

TI Designsリファレンス・デザイン 双方向DC/DCコンバータ



TI Designsリファレンス・デザイン

TI Designsリファレンス・デザインは、システムの迅速な評価とカスタム化に必要な方法、試験結果、設計ファイルなどを提供します。市場への投入期間短縮にも役立ちます。

デザイン・リソース

[TIDM-BIDIR-400-12](#)

デザイン・ファイルを含むツール・フォルダ



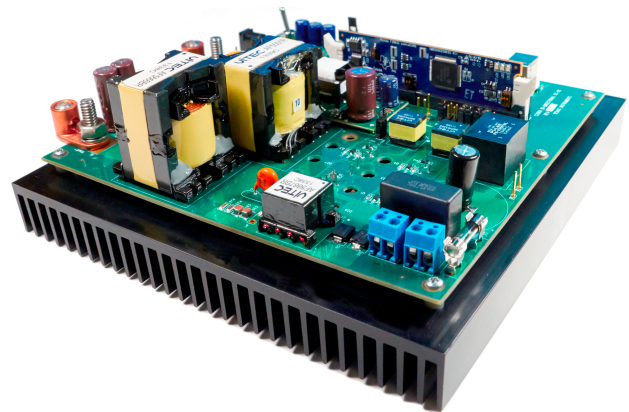
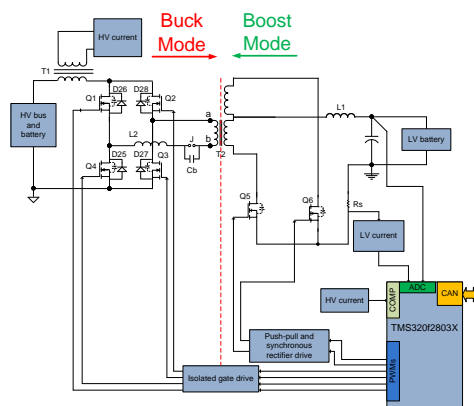
E2Eエキスパートに質問
WEBENCH®設計支援ツール

デザインの特長

- HVバス電圧範囲: 200V DC~400V DC
- LVバス電圧範囲: 9V DC~13.5V DC
- 各方向に300W定格の出力動作
- LVバスの定格電流33A、HVバスの定格電流1.8A
- 降圧と昇圧モード間のシームレスで迅速な切り替え
- 位相シフト・フルブリッジ動作(降圧モード)
- 電流給電プッシュプル動作(昇圧モード)
- スwitching周波数: 100kHz
- 出力インダクタ電流の電圧モード制御(VMC)および平均電流モード制御(ACMC)
- 複数の同期整流(SR)スイッチング方式
- 障害保護: 過電流、低電圧、過電圧

主なアプリケーション

- 車載用(HEV – ハイブリッド自動車、EV – 電気自動車)
- 汎用デジタル電源(産業用電源、バッテリー・バックアップ・システム)



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 概要

双方向DC-DCコンバータは、電力を双方向に伝達可能なアプリケーションで使用されます。ハイブリッド自動車 (HEV) および電気自動車 (EV) では、通常動作中はこれらの双方向コンバータによって低電圧 (12V) バッテリーが充電されますが (降圧モード)、高電圧 (400V/600V) バッテリーが非常に低いエネルギー/容量レベルまで放電したような緊急の状況では、高電圧バッテリーまたはバスを充電またはアシストします (昇圧モード)。標準的なシステムは、高電圧 (HV) 側のフルブリッジ電源段と、それに対して絶縁された低電圧 (LV) 側のフルブリッジ段または電流供給プッシュプル段から構成されています。

この実装では、LV側に配置されたTIの32ビット・マイコンTMS320F28035を使用して、両方向の電力の流れに対して閉ループ制御を実現しています。従来、これらのシステムでは、マイコンは監視または通信タスクだけを実行するように制限されてきました。高性能のマイコン・デバイスが利用できるようになったことで、マイコンはこれらのシステムの制御ループを完結させながら、従来のマイコン機能も処理できるようになりました。デジタル電源制御への移行は、以前はハードウェアで実装されていた機能がソフトウェアで実装されるようになったことを意味します。この能力は、柔軟性に加えて、システムの強化および単純化にも貢献しています。これらのシステムは、さまざまな条件下で電源段を最適に制御するための高度な制御戦略を実装しながら、システム・レベルのインテリジェンスを提供することで、動作モード間およびパルス幅変調 (PWM) スwitching・パターン間で安全かつシームレスな遷移を可能にします。

このドキュメントでは、このような絶縁型双方向DC-DCコンバータのマイコン・ベースの実装について詳しく説明します。位相シフト・フルブリッジ (PSFB) と同期整流によって、降圧モードでの400Vバス/バッテリーから12Vバッテリーへの電力フローを制御し、またプッシュプル段によって、昇圧モードでの低電圧バッテリーから高電圧バス/バッテリーへの逆電力フローを制御します。このデザインは、各モードで最大300Wの出力電力定格となっています。高電圧バスでの電圧は400V~200Vの範囲内であり、低電圧バスでの電圧は13.5V~9Vの範囲内です。両方の動作モードに対して、電圧モード制御と平均電流モード制御が実装されています。各種のPWMスitching方式が実装され、スitching・モード間、および2つの動作モード間でシームレスな遷移が行われます。

1.1 基本動作

降圧モード

PSFBコンバータは、絶縁型トランスの1次側のフルブリッジが形成する4個のパワー・エレクトロニクス・スイッチ (MOSFETまたはIGBTなど) と、2次側のダイオード整流器または同期整流用MOSFETスイッチによって構成されています。このトポロジによって、すべてのスitching・デバイスがゼロ電圧スitching (ZVS) でスitchingできるため、スitching損失の低減とコンバータの効率向上につながります。

このような絶縁型トポロジでは、2次側に信号整流が必要となります。低出力電圧または高出力電流の定格を持つシステムでは、同期整流を実装することで、ダイオード整流による損失を避けて最大の性能を達成できます。この構成では、各種のスitching方式により2次側に同期整流を実装することで、変化する負荷条件に基づき最適な性能を実現しています。

DC-DCコンバータ・システムは、電圧モード制御 (VMC)、平均電流モード制御 (ACMC)、またはピーク電流モード制御 (PCMC) などさまざまなモードで制御できます。同じ電源段を制御するためにこれらの異なる制御モードを実装する際には、一般に制御回路の再設計とともに電源段センシング回路にいくつかの変更が必要となります。マイコン・ベースのシステムにより、追加の変更なし、または最小限の変更のみで、これらすべてのモードを同じデザイン上で実験することができます。図 1 に、位相シフト・フルブリッジの概略回路図を示します。MOSFETスイッチ Q1、Q2、Q3、Q4 によって、T1 トランスの1次側にフルブリッジが形成されます。Q1 と Q4 は、50% のデューティ・サイクルで、互いに180度の位相差でスitchingされます。Q2 と Q3 は、50% のデューティ・サイクルで、互いに180度の位相差でスitchingされます。フルブリッジのレグ Q2-Q3 用の PWM スitching 信号は、レグ Q1-Q4 用の信号に対

して位相シフトされています。この位相シフトの大きさによって、対角線上のスイッチ間のオーバーラップの度合いが決まり、それによって転送されるエネルギーの量が決まります。D5とD6は2次側のダイオード整流を提供し、LoとCoは出力フィルタを形成します。インダクタLRは、MOSFETの容量との共振動作のためにトランスのリーク・インダクタンスをアシストし、ゼロ電圧スイッチング (ZVS) を容易にします。図 2 に、図 1 のシステムのスイッチング波形を示します。

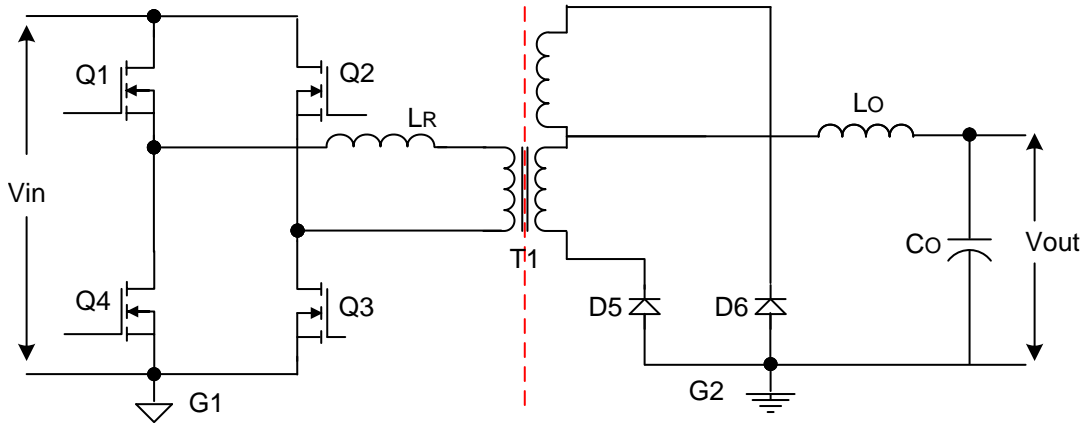


図 1. 降圧モードの電源段

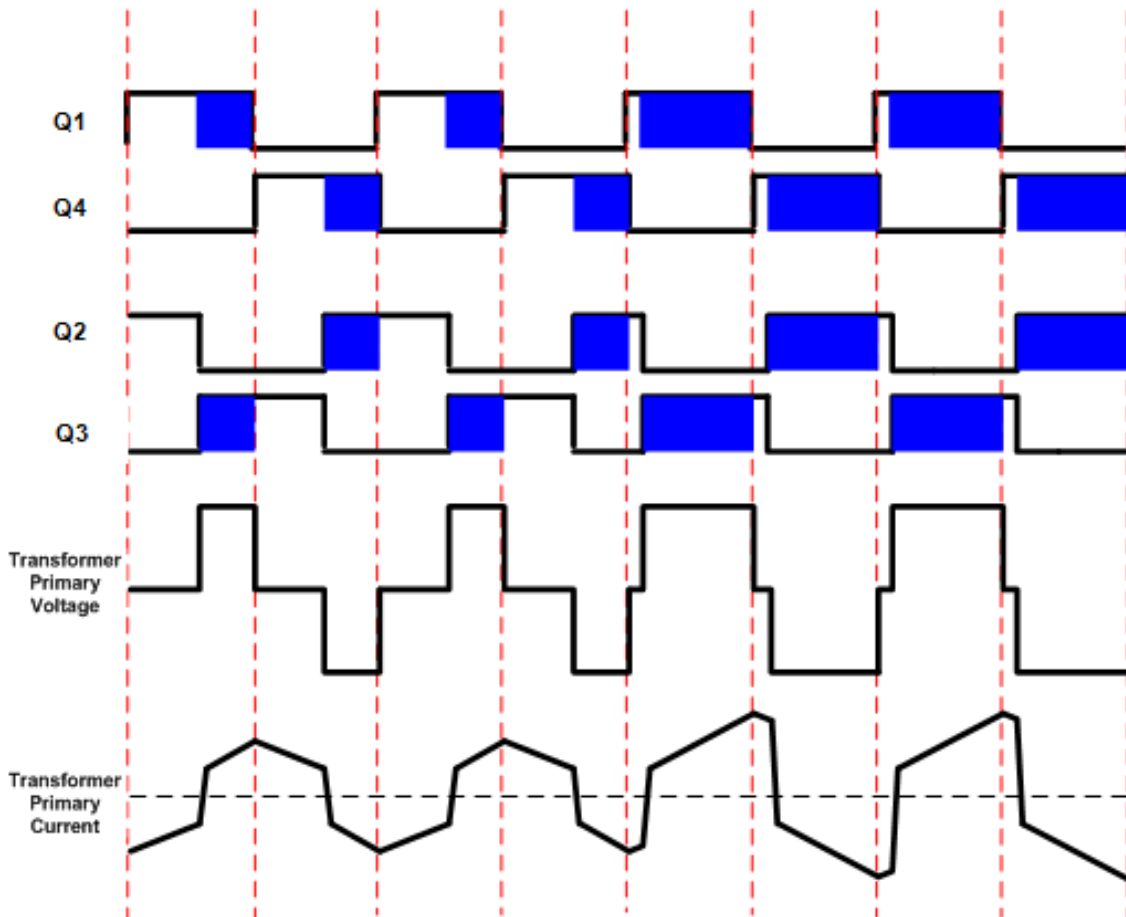


図 2. 降圧モードのPWM波形

昇圧モード

同期整流スイッチは、昇圧モードではプッシュプル・スイッチとなります。このモードでは降圧モード出力インダクタを電流源として使用できるため、このトポロジは電流給電のプッシュプル・コンバータとして動作できます。HV側のフルブリッジ・スイッチはオフに保持し、それらのボディ・ダイオードを整流に使用することができます。フルブリッジ・スイッチは、昇圧モードでのアクティブ整流に使用します。各プッシュプル・スイッチは、50%を超えるデューティ・サイクルのPWM信号で駆動され、それぞれ互いに180度位相がずれています。

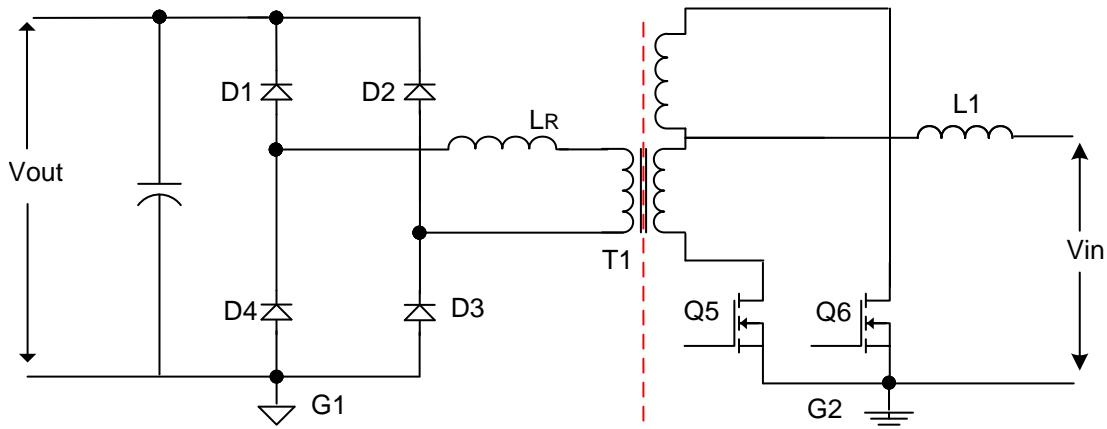


図 3. 昇圧モードの電源段

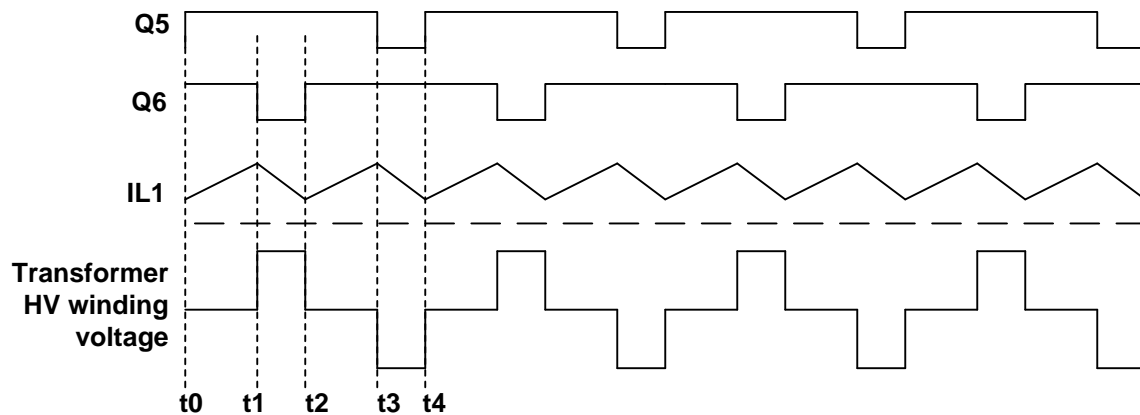


図 4. 昇圧モードのPWM波形

- **t0 – t1**: この期間中は、Q5とQ6が同時にオンになります。トランスの低電圧巻線の誘導性エネルギーおよび昇圧インダクタ(L1)の誘導性エネルギーが増加します。
- **t1 – t2**: t1の時点でQ6がオフになり、LV側に蓄積された誘導性エネルギーがダイオードD1およびD3を通してHV側に転送されます。

t2–t3期間中の動作はt0–t1と同じです。t3–t4期間中の動作はt1–t2と同様ですが、t3の時点でQ5がオフになり、HV側でD2とD4が導通します。

この動作モードでは、HV側に転送されるエネルギーの量は、スイッチQ5およびQ6を駆動する信号のデューティ・サイクルによって決まります。位相制御される降圧モードと異なり、これはデューティ制御による動作です。

1.2 双方向DC-DC基板上的実装

図 5 に、双方向DC-DC基板に実装された回路の概略ブロック図を示します。図 5 に示されるシステムでは、4 個のMOSFETスイッチ(Q1-Q4)が絶縁型トランスのHV側のフルブリッジを形成し、センタータップ付きのLV側の2 個のMOSFETスイッチ(Q5-Q6)が降圧モードでは同期整流器として、昇圧モードではプッシュプル・スイッチとして機能します。フルブリッジMOSFETのみを利用する昇圧モードの動作では、(整流用の)ボディ・ダイオードによってシステムの効率が大きく低下します。これは、ダイオードの逆方向回復が遅いことと、大きな電流が循環することによります。この欠点を解消するために、各フルブリッジ・スイッチにはショットキー・ダイオード(D25-D28)がアンチ・パラレル構成で接続されています。フルブリッジMOSFETは適切なタイミングでオン/オフされることにより、昇圧モードで同期整流を実現し、このモードでのシステム効率をさらに高めます。

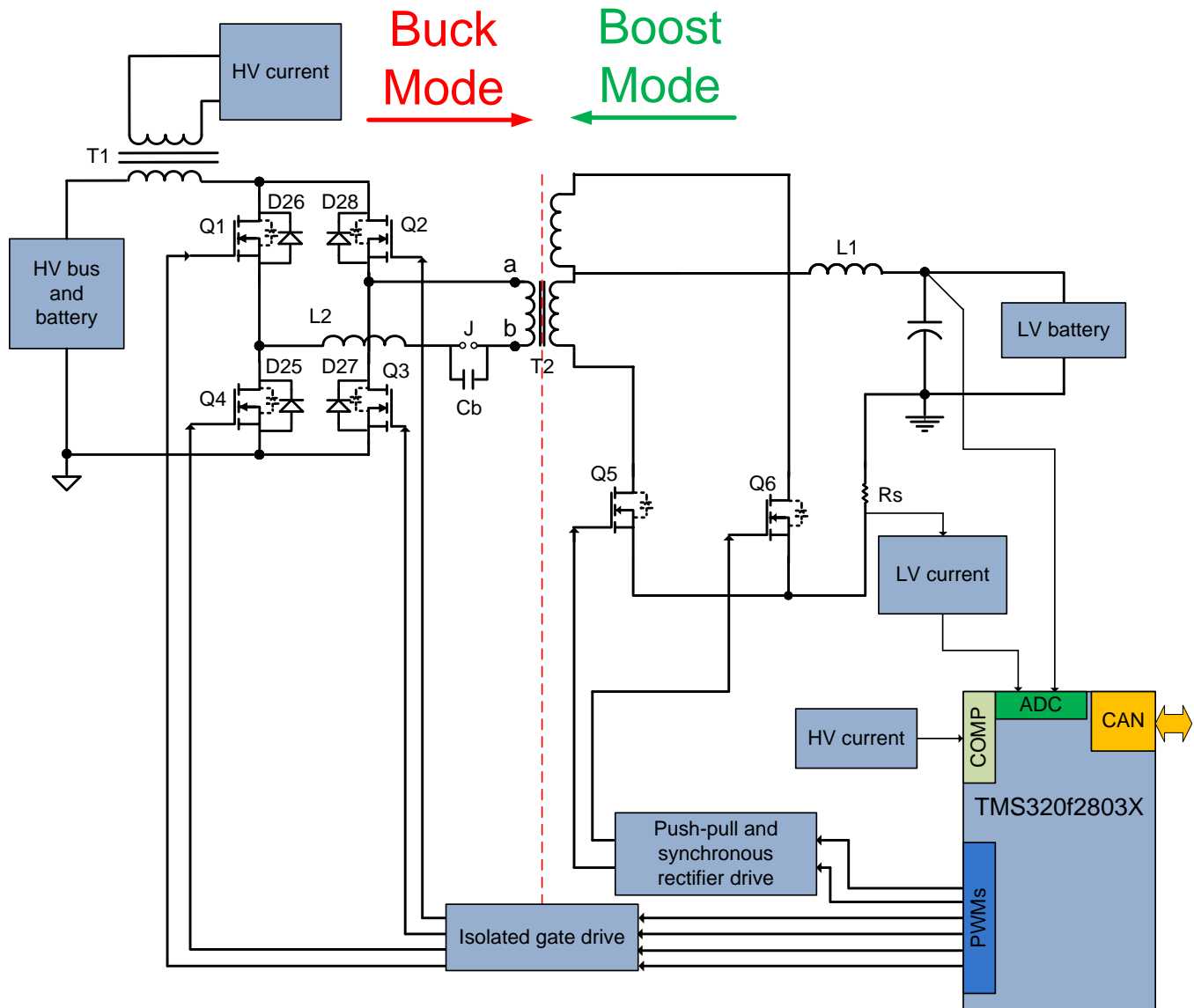


図 5. システム・ブロック図

制御アルゴリズムは、C2000™マイコン(MCU)に実装されています。このマイコンは、帰還信号とPWM出力を使用して電源段とやり取りします。コントローラは、このデザインのLV側に配置されます。絶縁型DC-DCシステムを設計する際の重要な手順の1つは、絶縁境界の位置を基準としたコントローラの配置を決めることです。コントローラをLV側に配置すると、複数のレールのあるシステムや、LV側で多くの信号や制御ループを処理するシステム、またはアプリケーションで(LV側の)他のシステムと通信する場合などに有効です。

このシステムを各種動作モードで制御するには、高速で高効率の制御ループ計算とともに、複雑なPWM駆動波形を生成する必要があります。この能力はC2000マイコン上でPWMモジュール、DACおよびスロープ補償ハードウェア付きのアナログ・コンパレータ、および高効率の32ビットCPUと組み合わされた12ビットの高速ADCなど、高度なオンチップ制御ペリフェラルによって可能になっています。以降のセクションでは、このソフトウェア・アルゴリズムについて詳しく説明します。

1.3 システムの特長

この実装の主な特長を次に示します。

- HVバス電圧範囲: 200V DC~400V DC
- LVバス電圧範囲: 9V DC~13.5V DC
- 各方向に300W定格の出力動作
- LVバスの定格電流33A、HVバスの定格電流1.8A
- 降圧と昇圧モード間のシームレスで迅速な切り替え
- 位相シフト・フルブリッジ動作(降圧モード)
- 電流給電プッシュプル動作(昇圧モード)
- スイッチング周波数: 100kHz
- 出力インダクタ電流のVMCとACMC
- 複数のSRスイッチング方式
- 障害保護: 過電流、低電圧、過電圧

2 ハードウェアとリソース

図 6 に、ハードウェアの主要なコンポーネントを示します。

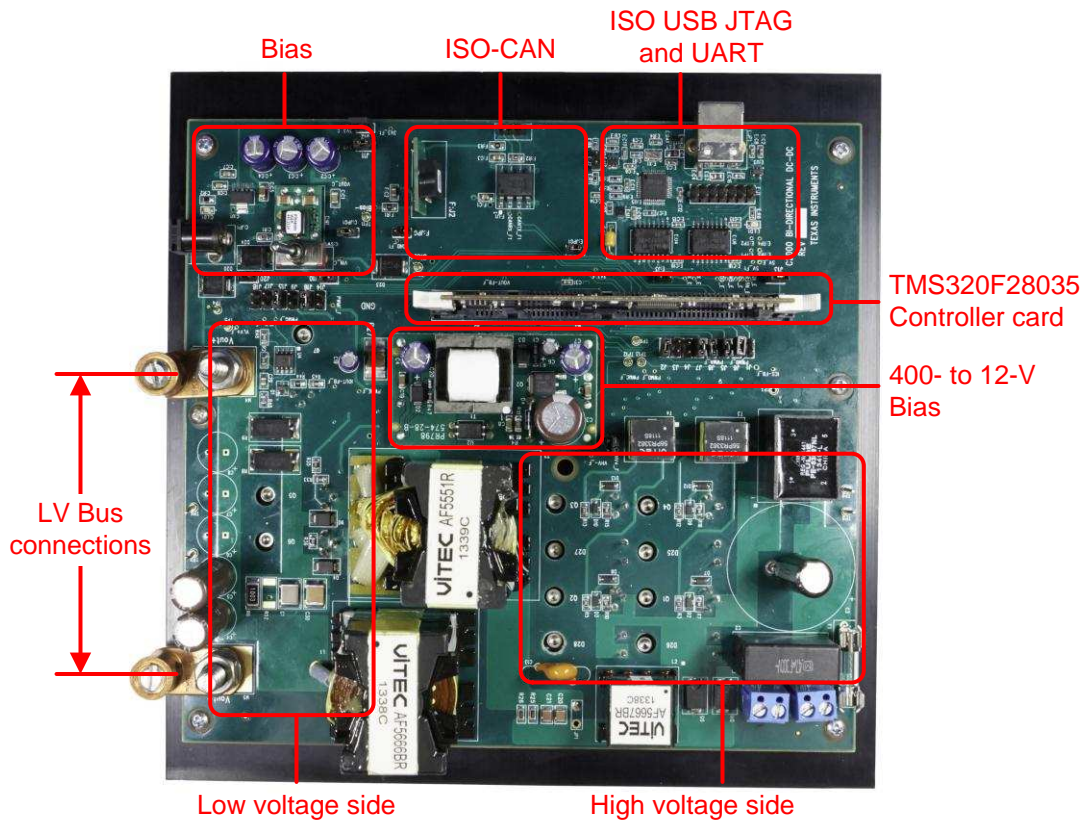


図 6. 双方向DC/DC基板

表 1に、F28035 controlCARDとベース基板との間の主要な信号接続を示します。また、図 7に回路図の関連部分を示しています。

表 1. 主要な信号接続

信号名	説明	controlCARDへの接続
ePWM-1A	フルブリッジ・スイッチQ2のPWM駆動	GPIO-00
ePWM-1B	フルブリッジ・スイッチQ3のPWM駆動	GPIO-01
ePWM-2A	フルブリッジ・スイッチQ1のPWM駆動	GPIO-02
ePWM-2B	フルブリッジ・スイッチQ4のPWM駆動	GPIO-03
ePWM-4A	同期整流器/ブッシュプル・スイッチQ6のPWM駆動	GPIO-07
ePWM-4B	同期整流器/ブッシュプル・スイッチQ5のPWM駆動	GPIO-06
VLV-FB	低電圧バス – 電圧帰還	ADC-A0
ILV-FB	低電圧電流帰還(ジャンパJ2を接続)	ADC-B3/COMP2A
ILV-FILT	大幅にフィルタリングされた低電圧電流帰還	ADC-B2
VHV-FB	高電圧バス – 電圧帰還	ADC-B1
IHV-FB	トランスの高電圧巻線電流	ADC-A2/COMP1A
IHV-FILT	大幅にフィルタリングされたトランスの高電圧巻線電流	ADC-A1

この基板には実験用の各種ジャンパ・オプションがありますが、基板を動作させるにはそのうちのいくつかを接続する必要があります。接続が必要なジャンパは以下のとおりです。

- C:JPG1
- E:JPG1
- J2
- J5
- J6
- J7
- J8
- J10
- J11
- J12
- J13
- J15
- J16
- J19

3 テスト結果

図 8 に、この基板をテストするためのテスト構成を示します。DC電源はバッテリーからではなくHVおよびLVバスから供給されています。150V、60Aのダイオード (GeneSiC Semiconductor, 1N2130A) をLVバスへの低電圧電源に接続し、600V、6Aのダイオード (Vishay, G1756) をHVバスへの高電圧電源に接続しています。LVバス上には電子負荷が使用され、HVバス上には抵抗性負荷バンクが使用されています。

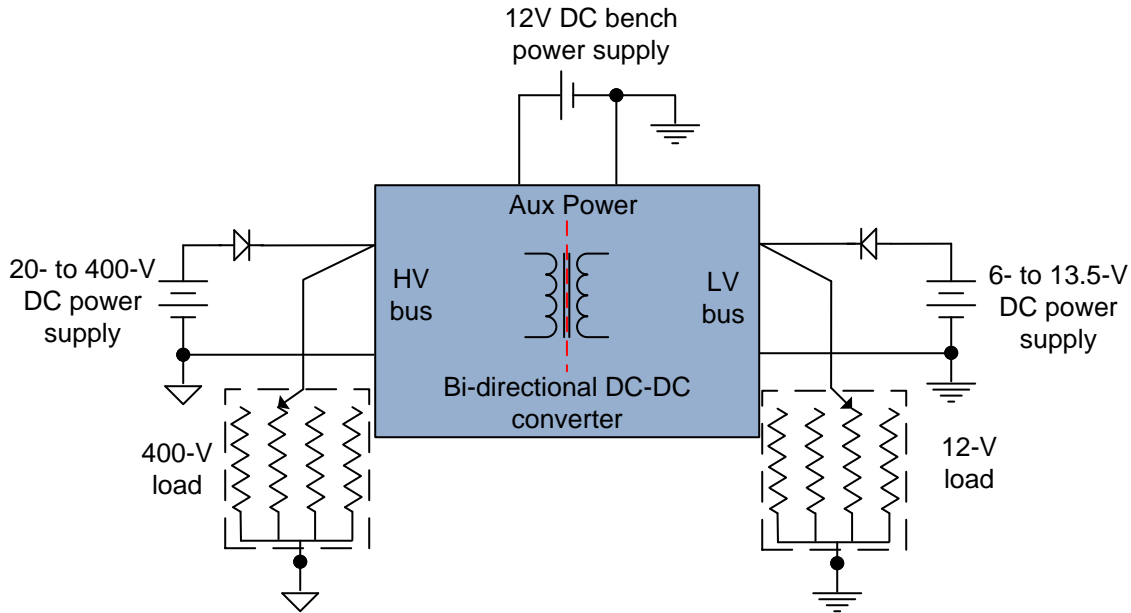


図 8. テスト構成

以降の図に、この基板を使用したいくつかの結果を示します。

降圧モード

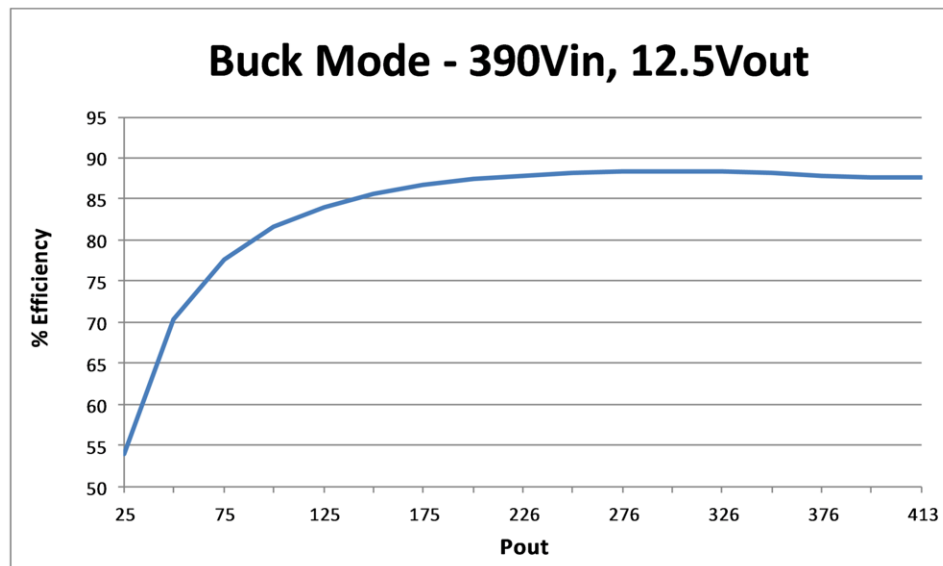


図 9. 効率 対 負荷 (390 Vinでの降圧モード)

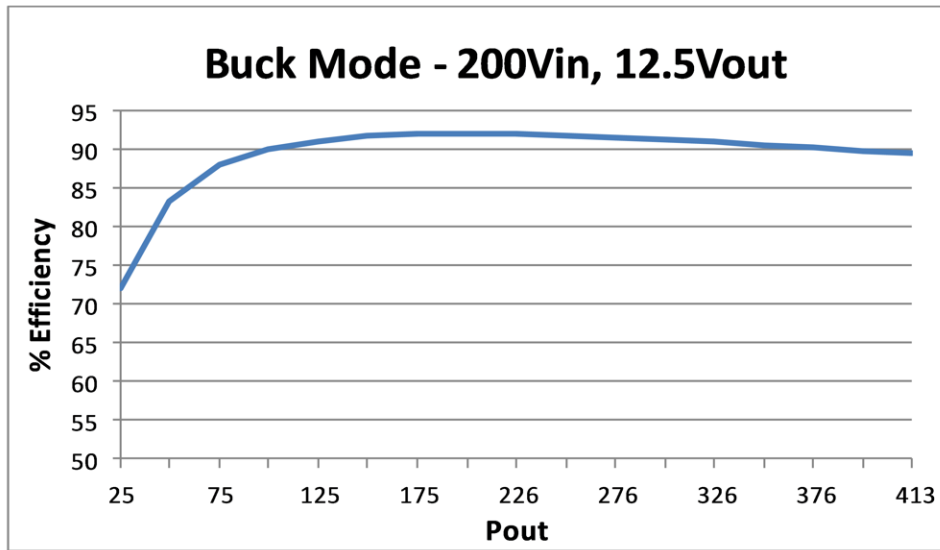


図 10. 効率 対 負荷 (200 Vinでの降圧モード)

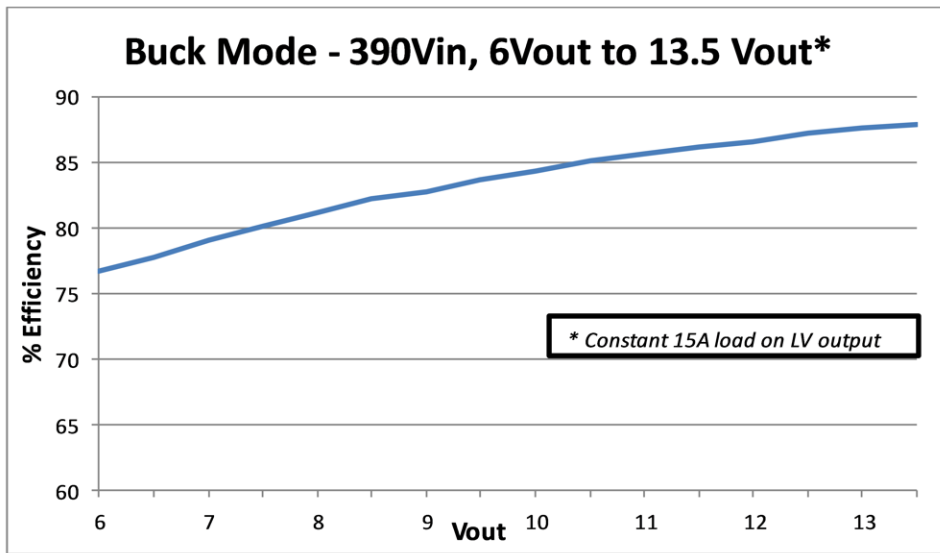


図 11. 効率 対 出力電圧 (390 Vinでの降圧モード)

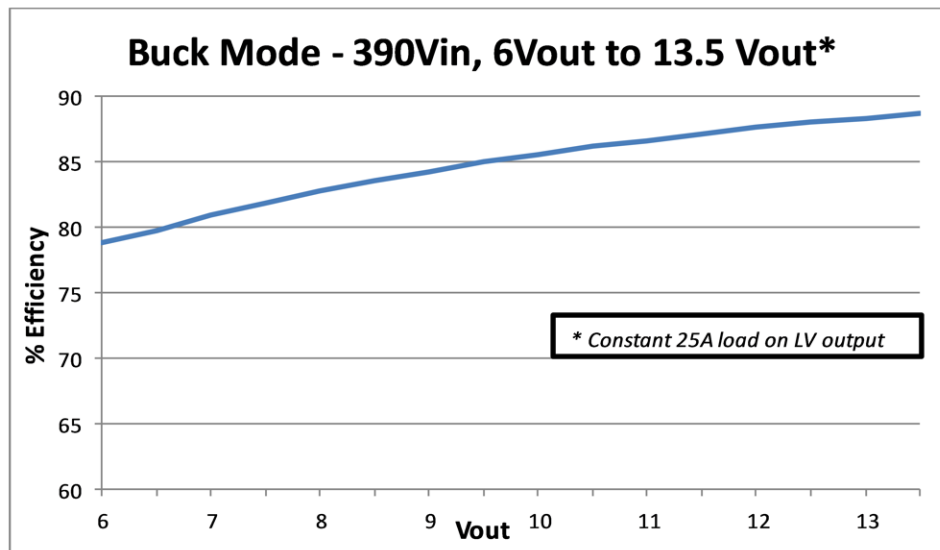


図 12. 効率 対 出力電圧 (390 Vinでの降圧モード)

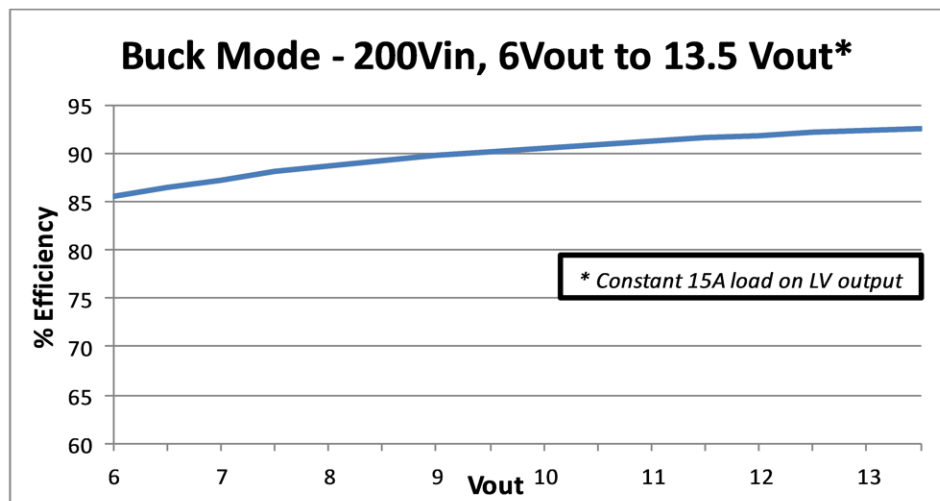


図 13. 効率 対 出力電圧 (200 Vinでの降圧モード)

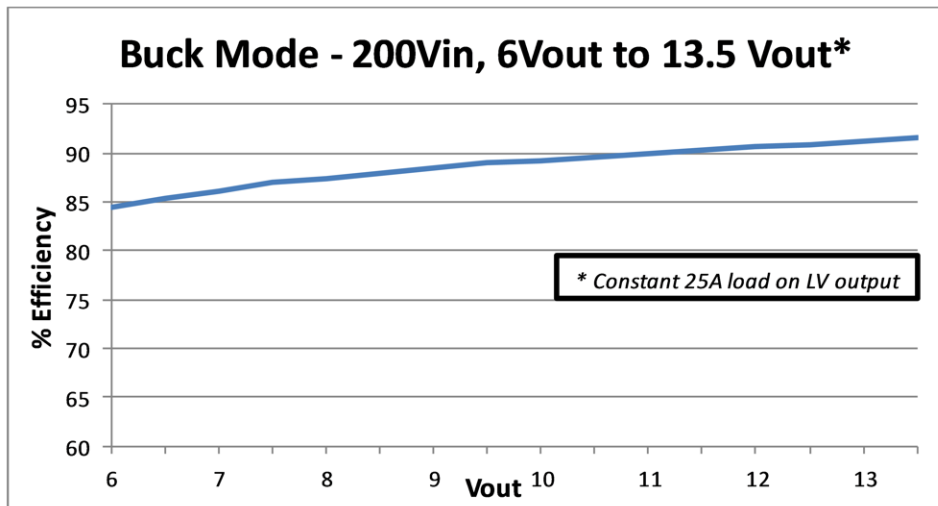


図 14. 効率 対 出力電圧 (200 Vinでの降圧モード)

昇圧モード

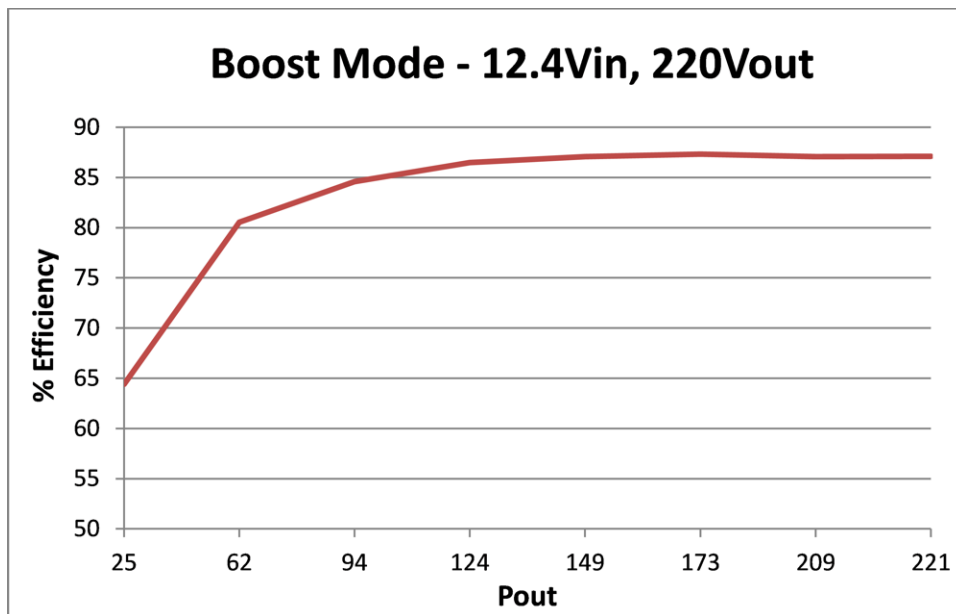


図 15. 効率 対 負荷 (12.4 Vin、220 Voutでの昇圧モード)

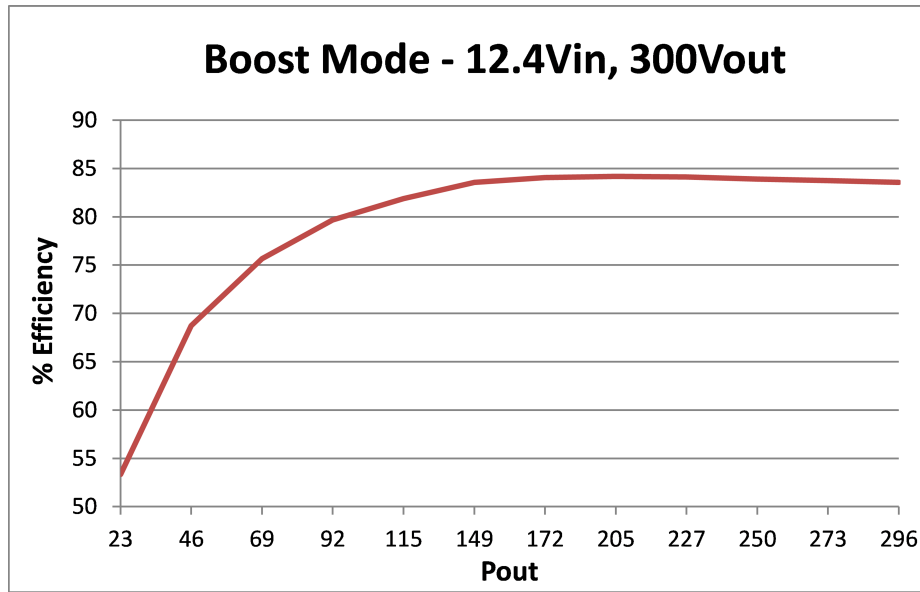


図 16. 効率 対 負荷 (12.4 Vin、300 Voutでの昇圧モード)

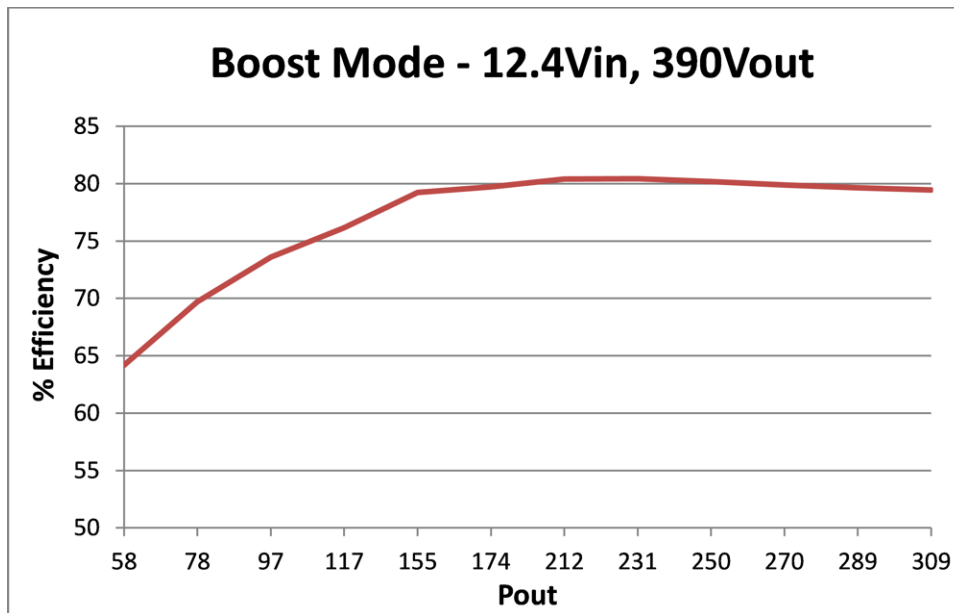


図 17. 効率 対 負荷 (12.4 Vin、390 Voutでの昇圧モード)

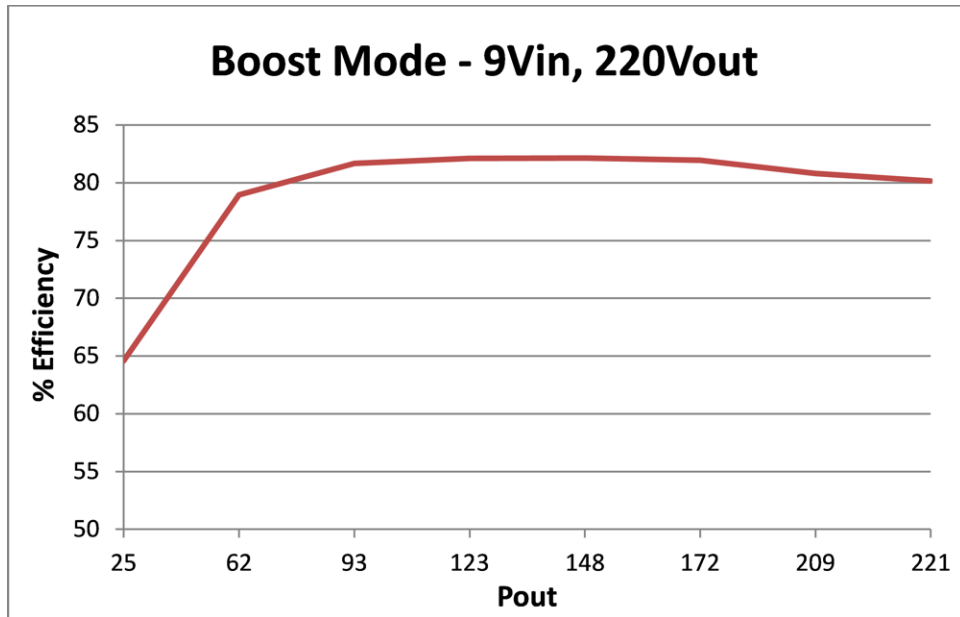


図 18. 効率 対 負荷(9 Vin、220 Voutでの昇圧モード)

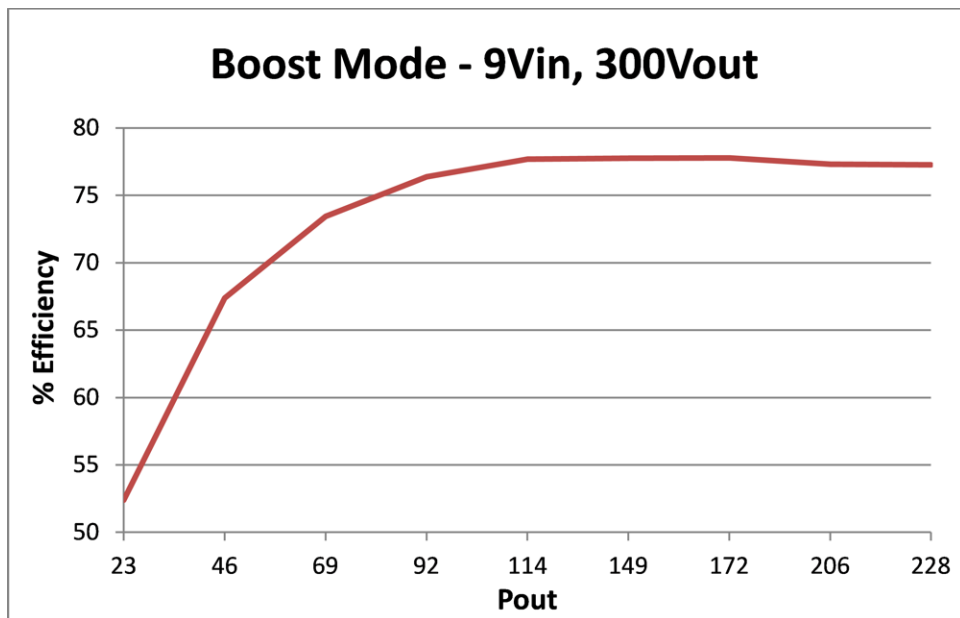


図 19. 効率 対 負荷(9 Vin、300 Voutでの昇圧モード)

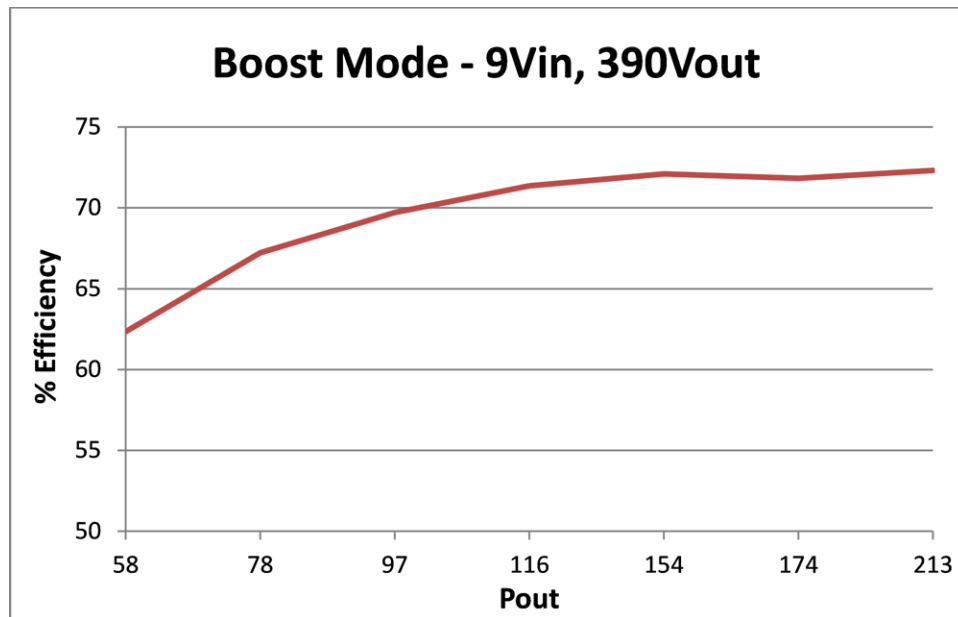


図 20. 効率 対 負荷(9 Vin、390 Voutでの昇圧モード)

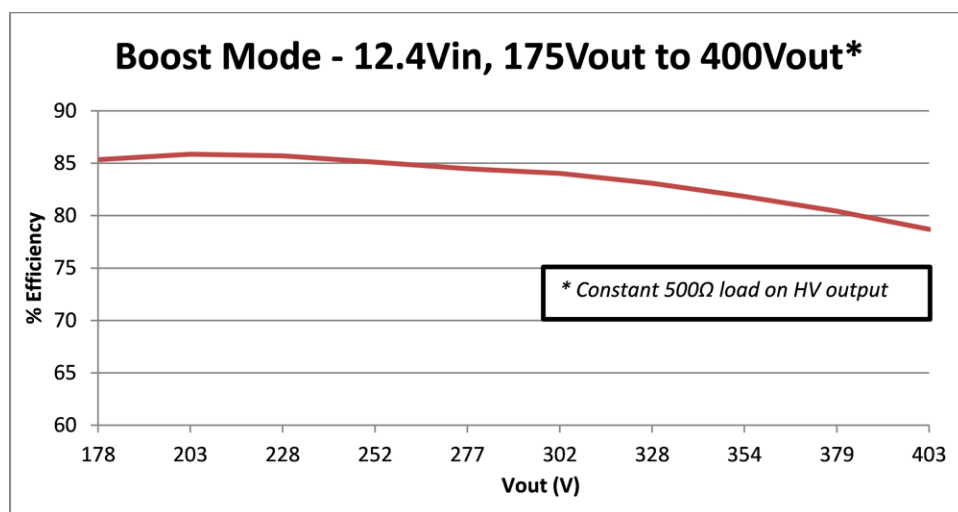


図 21. 効率 対 出力電圧(12.4 Vinでの昇圧モード)

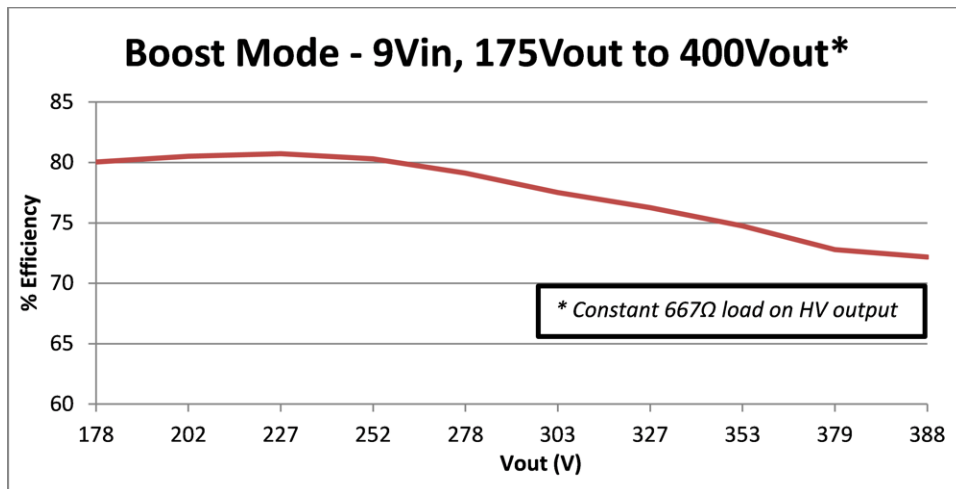


図 22. 効率 対 出力電圧 (9 Vinでの昇圧モード)

モード遷移 (V_{LV} 、 V_{HV} 電圧)

以降の図に、各種の動作条件でのモード遷移中に高電圧バスおよび低電圧バスで観測される電圧波形を示します。新しいモードでの電圧設定点(モード変更コマンド後)は、そのバス上の前の電圧よりも高い点に設定されます。たとえば、降圧モードから昇圧モードに遷移するときには、HVバスの電圧 (V_{HV}) が、前の V_{HV} 電圧よりも4V高い値に設定されます。昇圧モードから降圧モードに遷移するときは、 V_{LV} 電圧が前の V_{LV} 電圧よりも0.5V高い値に設定されます。結果として、モード遷移中の2つのバスには、予想どおりの電圧の低下または上昇が見られています。

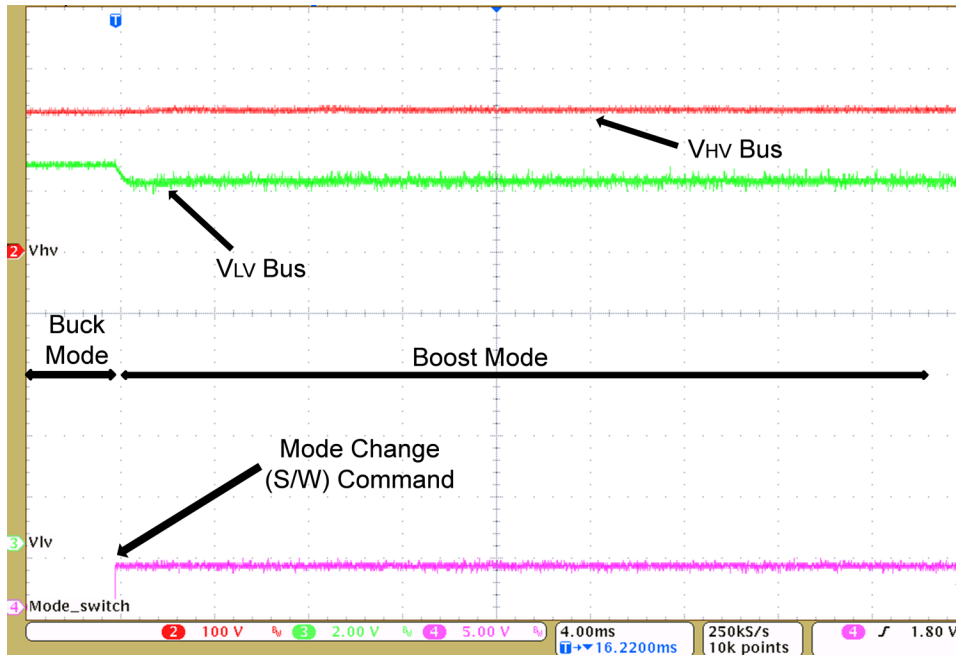


図 23. 降圧から昇圧へのモード遷移 ($V_{HV} = 225V$ 、 $2k\Omega$ 負荷、 $V_{LV} = 12.6V$ 、 3Ω 負荷)

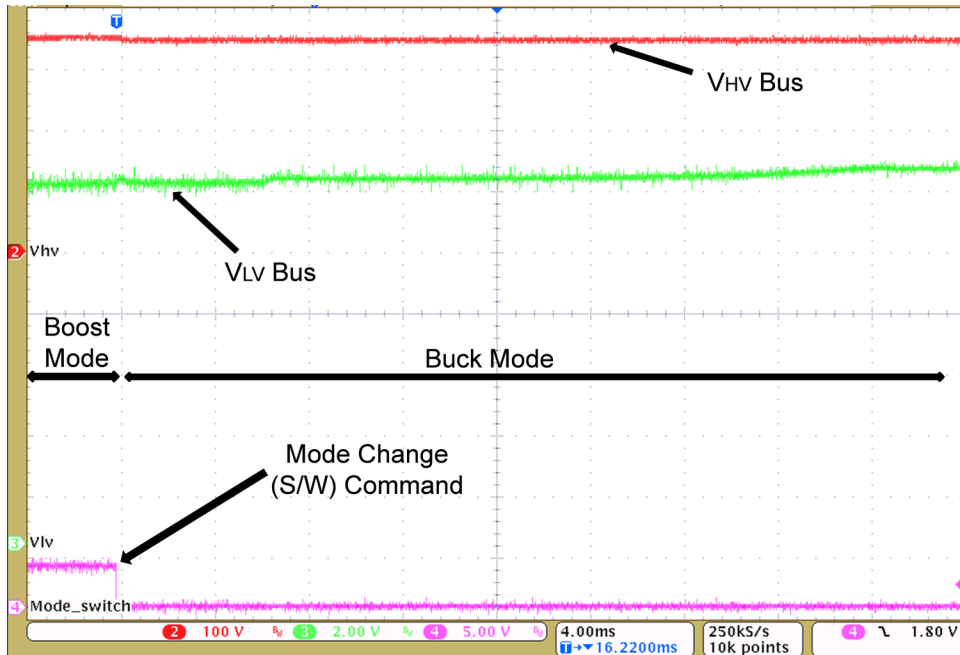


図 26. 昇圧から降圧へのモード遷移 ($V_{HV} = 350V$ 、 $2k\Omega$ 負荷、 $V_{LV} = 12.6V$ 、 10Ω 負荷)

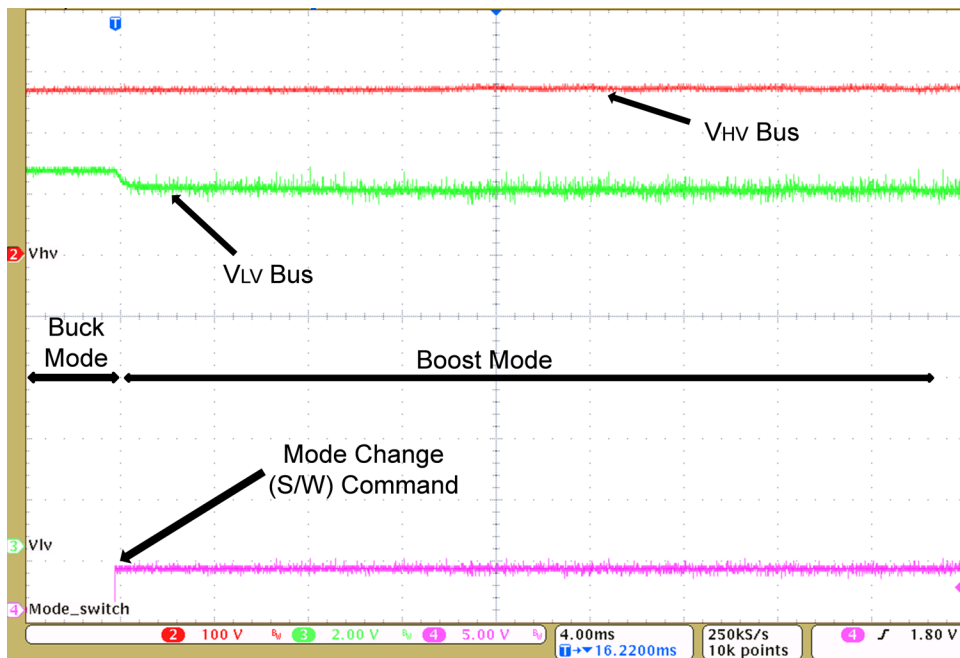


図 27. 降圧から昇圧へのモード遷移 ($V_{HV} = 266V$ 、 667Ω 負荷、 $V_{LV} = 12.6V$ 、 10Ω 負荷)

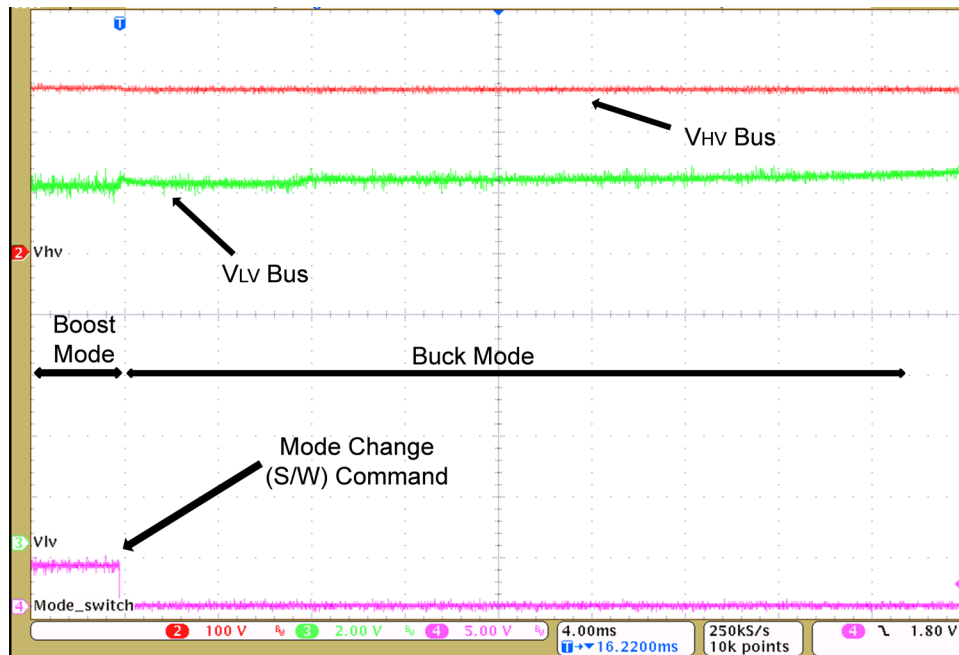


図 28. 昇圧から降圧へのモード遷移 ($V_{HV} = 266V$ 、 667Ω 負荷、 $V_{LV} = 12.6V$ 、 10Ω 負荷)

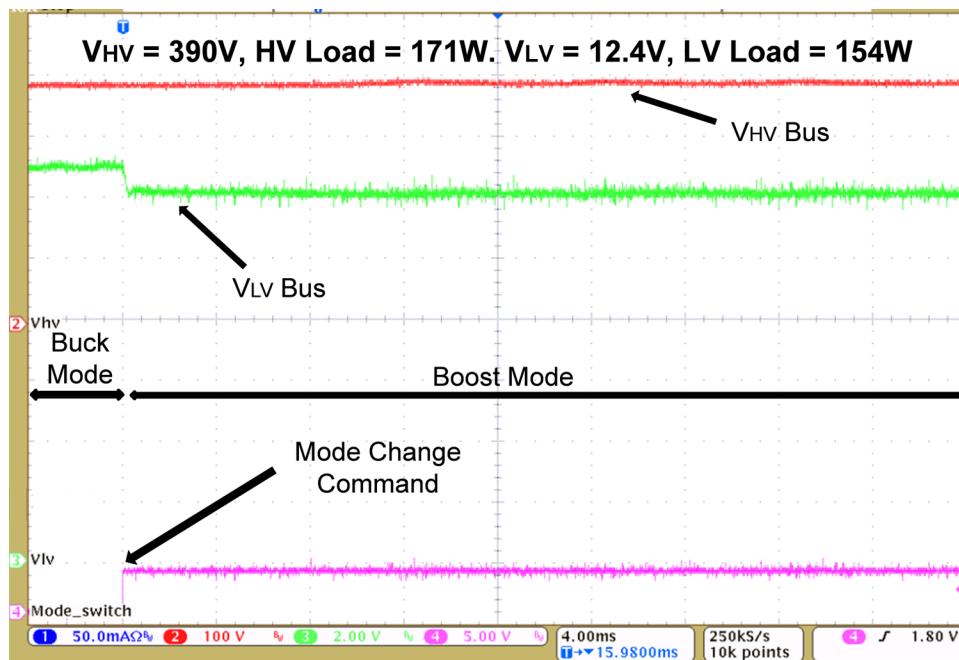


図 29. 降圧から昇圧へのモード遷移 ($V_{HV} = 390V$ 、 $171W$ 負荷、 $V_{LV} = 12.4V$ 、 $154W$ 負荷)

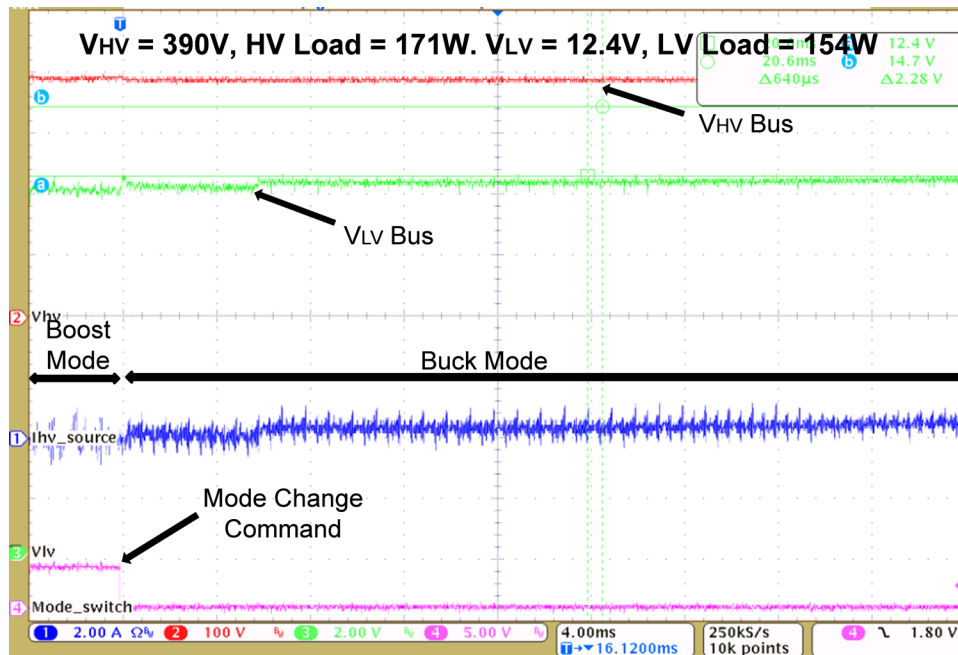


図 30. 昇圧から降圧へのモード遷移 ($V_{HV} = 390V$ 、 $171W$ 負荷、 $V_{LV} = 12.4V$ 、 $154W$ 負荷)

モード遷移(拡大)

以降の図には、各種動作条件でのモード遷移中にメインの電源トランスT2の高電圧巻線で観測された電圧波形、および対角線上に配置されたフルブリッジ・スイッチQ2およびQ4のPWM駆動信号について、拡大図を示します。条件

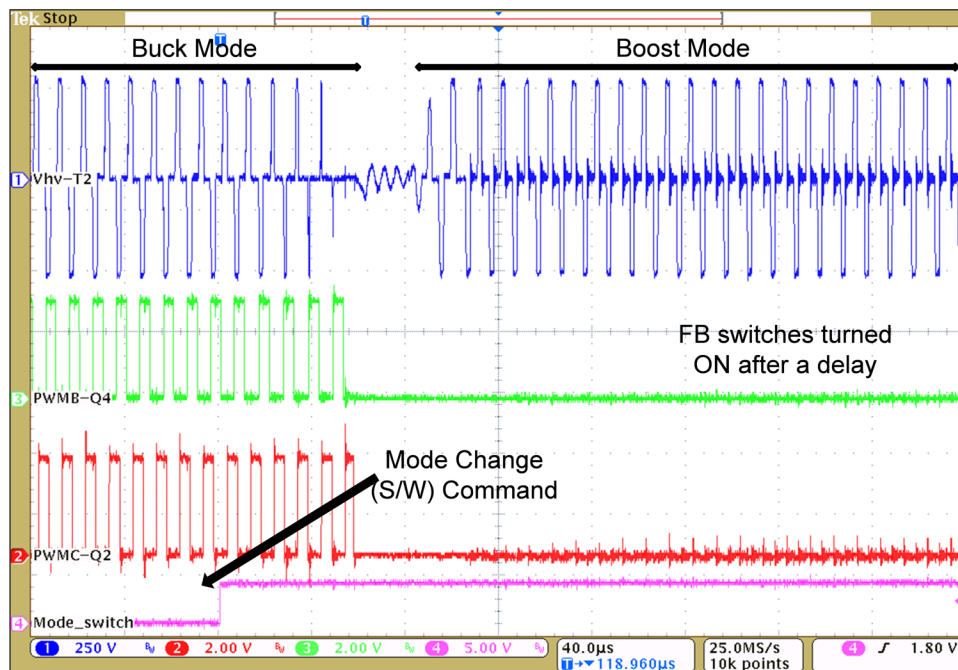


図 31. 降圧から昇圧へのモード遷移 ($V_{HV} = 390V$ 、 $19W$ 負荷、 $V_{LV} = 12.4V$ 、 $307W$ 負荷)

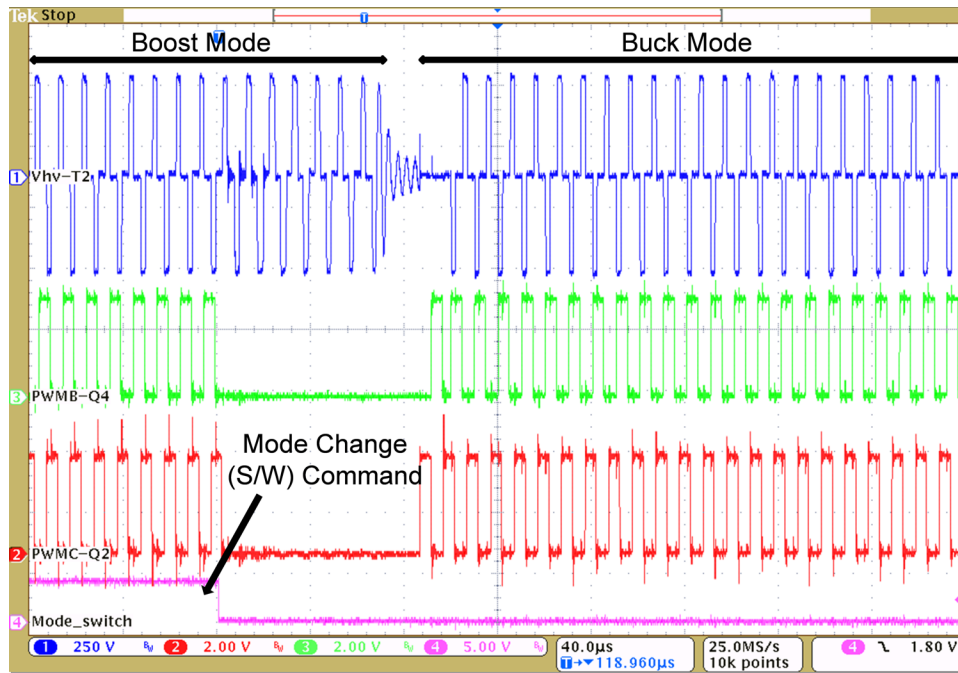


図 32. 昇圧から降圧へのモード遷移 ($V_{HV} = 390V$ 、 $19W$ 負荷、 $V_{LV} = 12.4V$ 、 $307W$ 負荷)

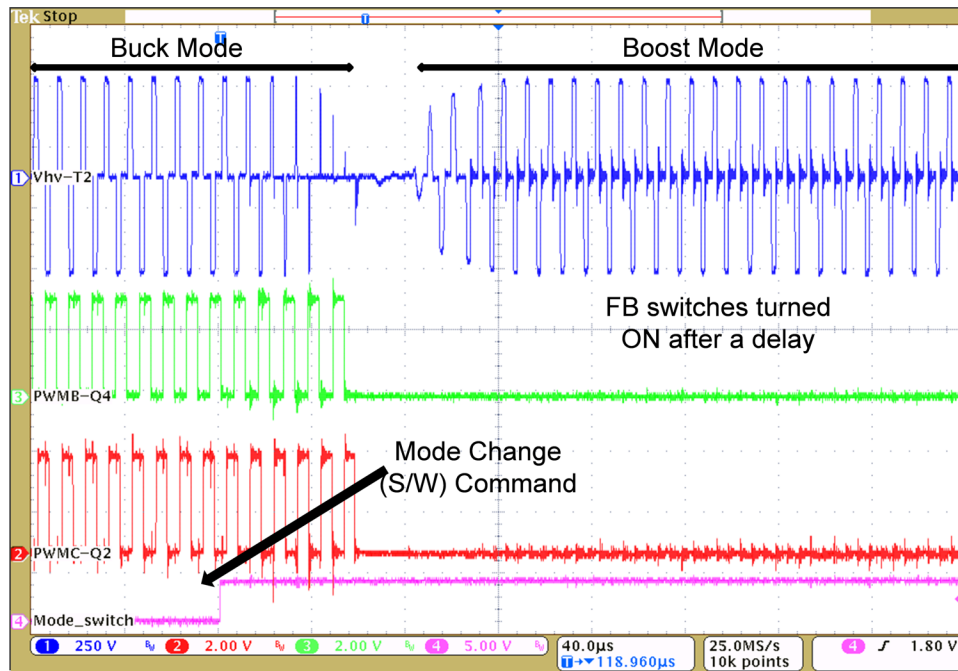


図 33. 降圧から昇圧へのモード遷移 ($V_{HV} = 390V$ 、 $171W$ 負荷、 $V_{LV} = 12.4V$ 、 $154W$ 負荷)

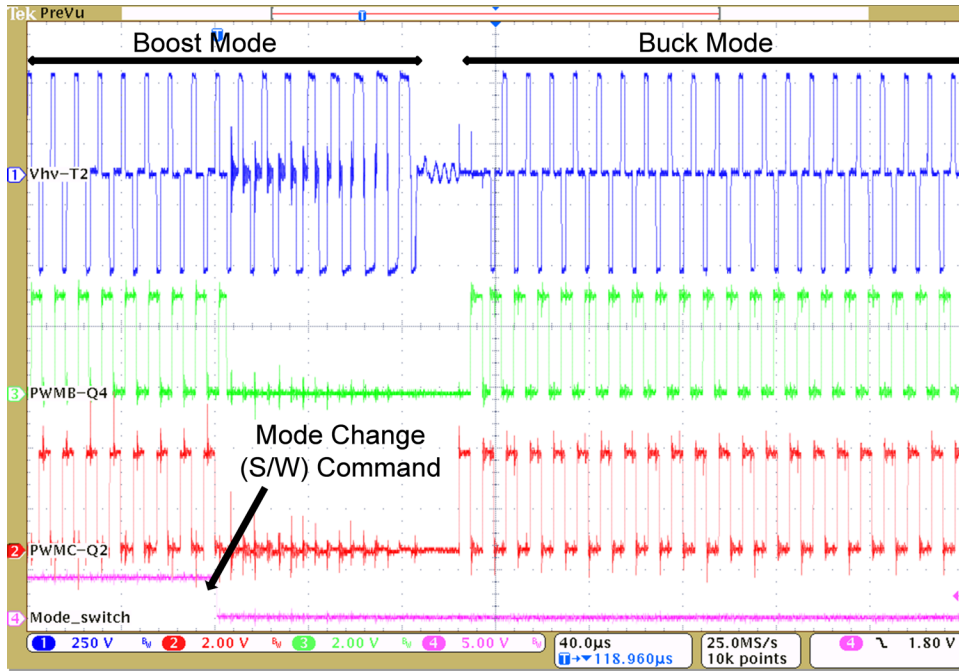


図 34. 昇圧から降圧へのモード遷移 ($V_{HV} = 390V$ 、 $171W$ 負荷、 $V_{LV} = 12.4V$ 、 $154W$ 負荷)

4 降圧モード(位相シフト・フルブリッジ) – 機能説明

4.1 電圧モード制御

VMC実装では、各レグのスイッチを固定(50%)デューティ・サイクルおよび固定周波数の相補型PWM信号で駆動します。図 35に示すように、コントローラは、ブリッジの1つのレグのスイッチを駆動するPWM信号を直接駆動し、位相シフトを制御します。位相シフトは、ブリッジのもう一方のレグのスイッチを駆動する信号を基準に制御します。この位相シフトによって、図 37に示すように、対向するスイッチ間のオーバーラップの大きさが決まります。対向スイッチ間のオーバーラップが大きいほど、入力電圧がトランスの高電圧巻線に印加される時間が長くなり、低電圧側に転送されるエネルギーが増加します。コントローラは、2つのフルブリッジ・レグを駆動するPWM信号間の位相シフトを直接制御してこのエネルギー転送を制御することにより、出力のレギュレーションを実現します。この位相シフトが、制御パラメータです。VMC実装では、時間の経過に伴う磁束の不均衡によって起こりうるトランスの飽和を避けるために、トランスの高電圧巻線にDCブロッキング・コンデンサを含める必要があります。ジャンパJP1は、デフォルトでは接続されません。PCMCを実装する場合は、適切な定格のジャンパをJP1に取り付けることができます。

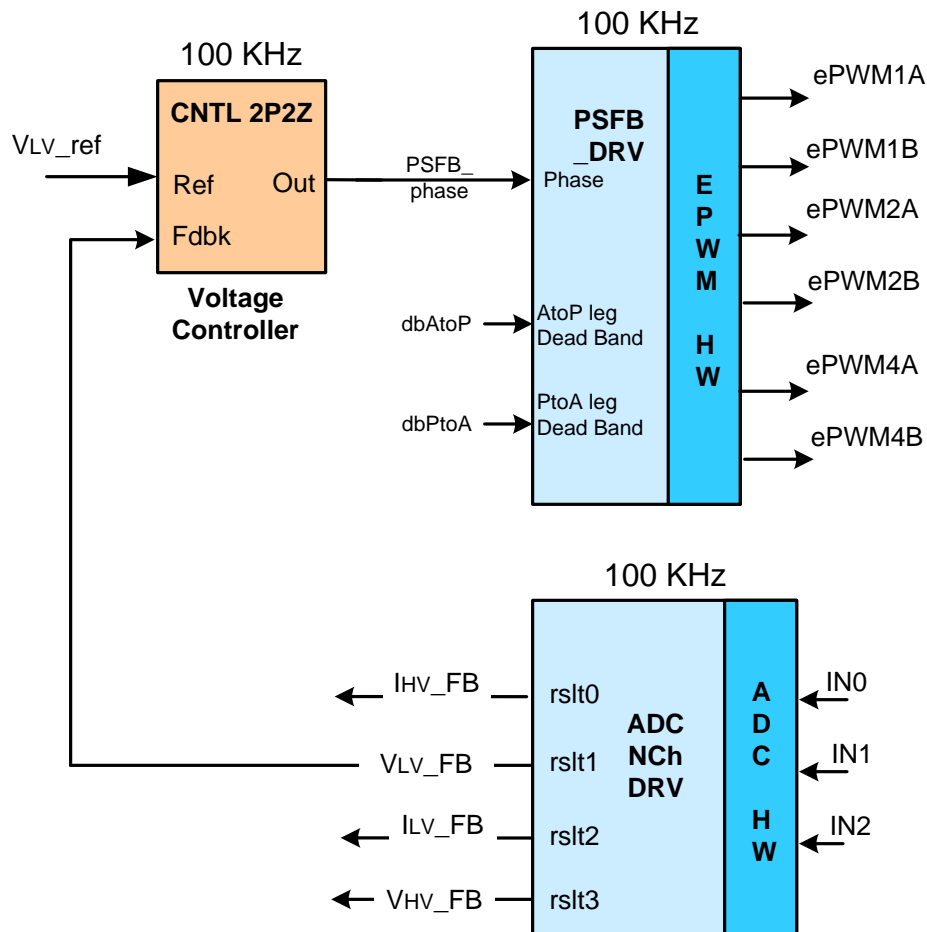


図 35. VMCのブロック図

4.2 出カインダクタ電流の平均電流モード制御

電源アプリケーションでの標準的なACMC実装は、外側の電圧制御ループと内側の平均電流ループから構成されます。PSFB電源段に対する内側の平均電流ループは、高電圧巻線の平均電流を制御します。バッテリー充電アプリケーションでは、1つの平均電流ループを使用してACMCモードのバッテリー充電電流を制御し(定電流充電)、必要に応じて定電圧充電用にVMCモードへと切り替えます。ACMCの実装は、波形生成および電源段制御についてはVMCの実装と同様です。図36に示すように、コントローラは、ブリッジの1つのlegのスイッチを駆動するPWM信号を直接駆動し、位相シフトを制御します。位相シフトは、ブリッジのもう一方のlegのスイッチを駆動する信号を基準に制御します。VMCモードと同様に、コントローラは、2つのフルブリッジ・legを駆動するPWM信号間の位相シフトを直接制御してエネルギー転送を制御することにより、出力のレギュレーションを実現します。図37に、降圧モード動作中のさまざまな波形を示します。

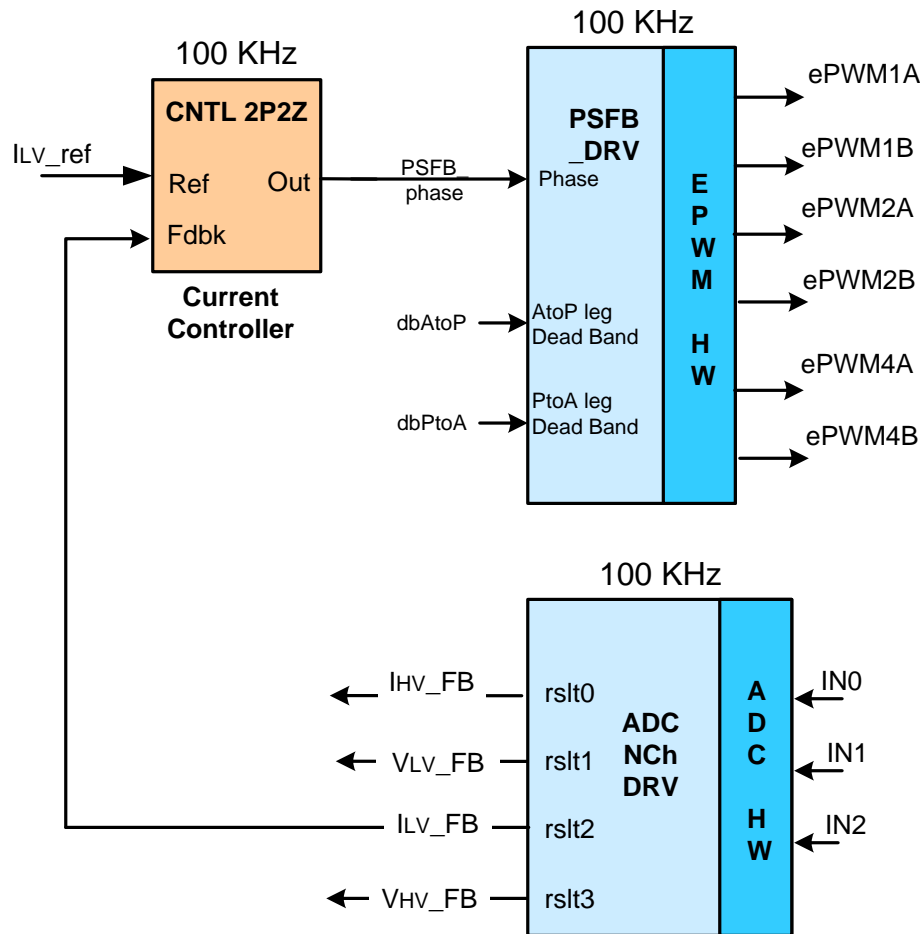


図 36. ACMCのブロック図

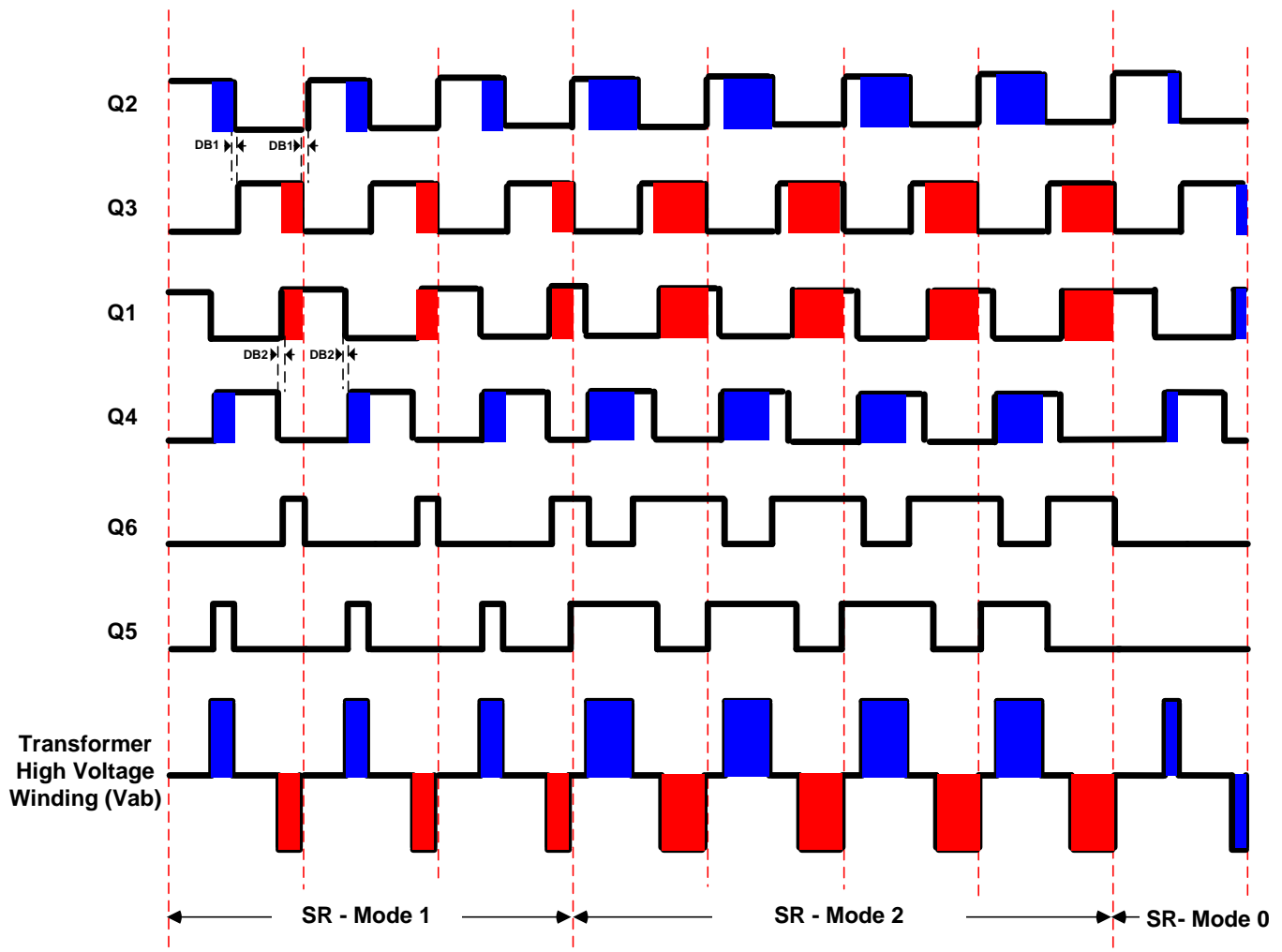


図 37. 降圧モードの波形

4.3 ゼロ電圧スイッチングまたは低電圧スイッチング

PSFB DC-DCコンバータは、回路内の寄生要素を使用して、MOSFETスイッチをオンにする前に印加されている電圧がゼロであるようにし、ソフトなスイッチングを実現します。このように電圧をかけないことで、ハードなスイッチングに伴うスイッチング損失を減らすことができます。

Q2-Q3レグのスイッチに対するスイッチング遷移によって、電力転送期間が終了します。このレグは、アクティブ・パッシブ・レグと呼ばれます。このレグのスイッチに対して遷移が発生すると、高電圧巻線の電流は、対応するPWMスイッチング・サイクルの半分の期間にわたって最大値に近くなります。反射された負荷電流が、この期間中の高電圧巻線回路の循環エネルギーの補助となるため、このレグのスイッチにかかる電圧を0Vに近づけることが可能になります。スイッチに対するZVSは、このQ2-Q3レグで幅広い負荷範囲にわたって実現可能です。負荷が小さくなると、ZVSを実現またはそれに近づくためにデッドタイムを長くする必要があります。

Q1–Q4レグのスイッチに対するスイッチング遷移によって、電力転送期間が開始されます。このレグは、パッシブ・アクティブ・レグと呼ばれます。これらの遷移中は、高電圧巻線の電流が減少します。高電圧巻線電流はゼロ電流値を横切り、方向が変化します。電流がゼロになって方向が変わることで、ZVSのために利用できるエネルギーが小さくなります。低負荷状態で動作する場合は、これらのスイッチがオンになる前に、スイッチにかかる電圧がゼロにならなくてもかまいません。これらにかかる電圧がLVSを実現するための最小値であれば、これらのスイッチを一度にオンにすることで、スイッチング損失を最小限に抑えることができます。負荷が変化すると、LVSを実現するためにスイッチをオンにするタイミングが変化するため、Q2–Q3レグのスイッチと同様なデッドタイム調整が必要になります。

4.4 同期整流

同期整流器は、任意の時点で次の3つのモードのいずれかで動作できます。

- **モード0:** これは古典的なダイオード整流モードであり、同期整流器をオフに保持することで実現されます。これは、同期整流で得られる電力の節減よりも同期整流器のスイッチング損失の方が大きいような非常に低負荷の動作に対して有効です。
- **モード1:** このモードでは、同期整流器が理想的なダイオード動作をエミュレートします。このモードは、非常に低い負荷～比較的低い負荷での動作時、一般にはバースト・モードの使用時に便利です。このモードでは、対応する対向フルブリッジ・スイッチのペアを駆動するPWM信号が互いにオーバーラップするときだけ、同期整流器のMOSFETがオンになります。
- **モード2:** 他のすべての負荷条件に対して有効です。このモードでは、対応する対向フルブリッジ・スイッチとは反対側のペアを駆動するPWM信号が互いにオーバーラップするときだけ、同期整流器のMOSFETがオフになります。

図 37 に、これらのモードで同期整流器スイッチを駆動するために生成される波形を示します。システムの安全な動作を確保するために、大きな負荷過渡事象や急激な位相シフト変化コマンド時であっても、PWM出力にグリッチや異常なしでシームレスなモード遷移を実現できるようにすることが重要です。

5 降圧モード – ソフトウェアの概要

5.1 ソフトウェア制御フロー

Bi-dir_Buckプロジェクトでは、Cバックグラウンド/ASM-ISRフレームワークを使用します。Cバックグラウンド/ASM-ISRフレームワークは、アプリケーションのメインのサポート・プログラムとしてCコードを使用し、すべてのシステム管理タスク、意思決定、インテリジェンス、およびホストとのやり取りを実行します。アセンブリ・コードは、すべての重要な制御コードを実行する割り込みサービス・ルーチン(ISR)のみに厳密に制限されています。これには一般に、ADCの読み取り、制御用の計算、およびPWMとDACの更新が含まれます。図 38に、このプロジェクトの全般的なソフトウェア・フローを示します。

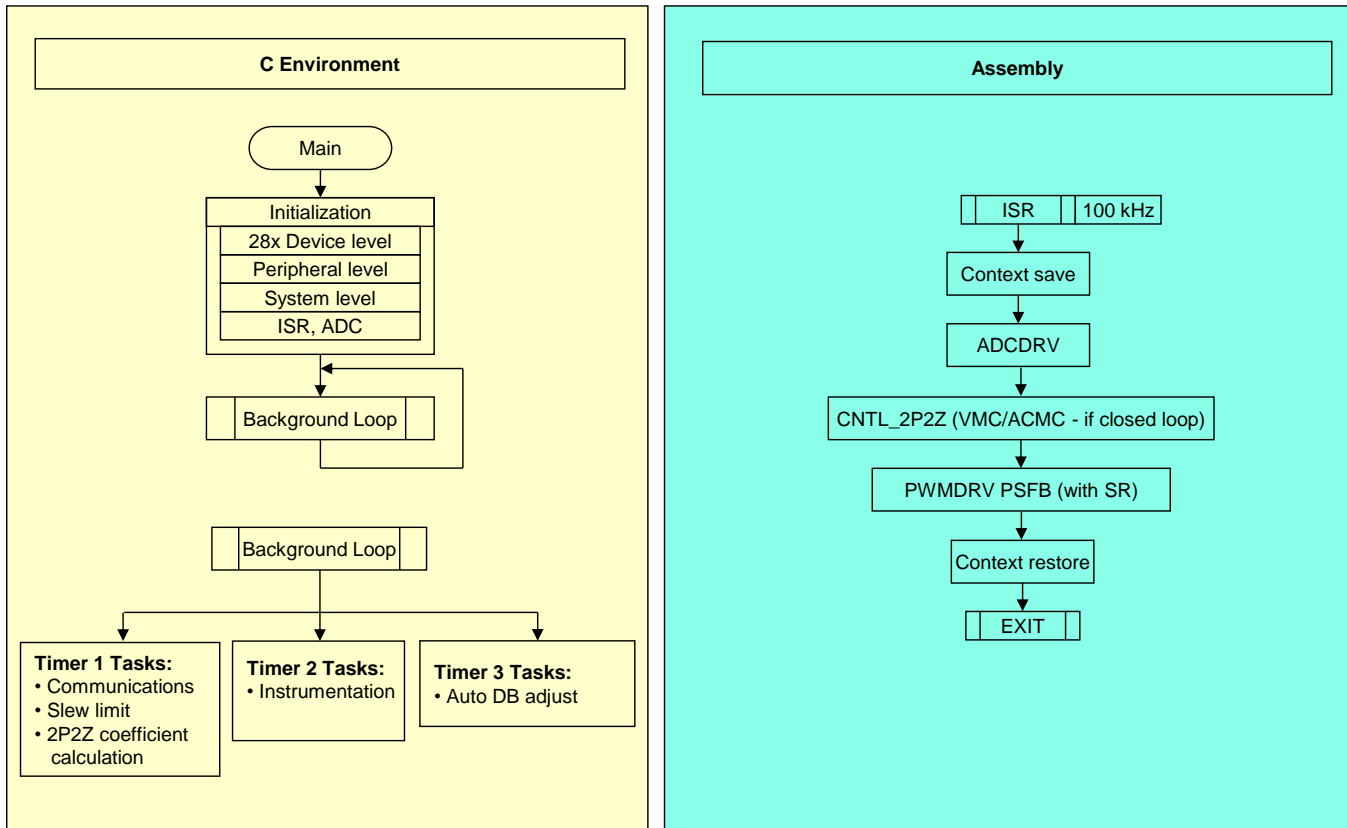


図 38. 降圧モードのソフトウェア・フロー

このプロジェクトでは、以下の主要なフレームワークCファイルが使用されます。

- **Bi-dir-Main.c** – このファイルはアプリケーションを初期化、実行、および管理します。
- **Bi-dir-DevInit_F2803x.c** – このファイルは、マイコンの1回だけの初期化および構成を管理し、クロック、PLL、GPIOの設定などの機能を含みます。

ISRは、以下のファイルから構成されます。

- **Bi-dir-DPL-ISR.asm** – このファイルは、時間が重要となる制御タイプ・コードをすべて含んでいます。このファイルには初期化セクション(1回実行)とランタイム・セクション(PWMスイッチング周波数で実行)が含まれています。

Power Libraryの関数(モジュール)は、このフレームワークから呼び出されます。

ライブラリ・モジュールにはCコンポーネントとアセンブリ・コンポーネントの両方が含まれる場合があります。表 2に、このプロジェクトのライブラリ・モジュールを示します。Cモジュール名および対応するアセンブリ・モジュール名を記載しています。

表 2. ライブラリ・モジュール

C構成関数	ASM初期化マクロ	ASMランタイム・マクロ
DAC_Cnf.c		
ADC_SOC_Cnf.c	ADCDRV_4CH_INIT m,n,p,q	ADCDRV_4CH m,n,p,q
PWM_PSFB_VMC_SR_Cnf.c	PWMDRV_PSFB_VMC_SR_INIT m,n,p	PWMDRV_PSFB_VMC_SR m,n,p
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

図 39 に制御ブロックを示します。

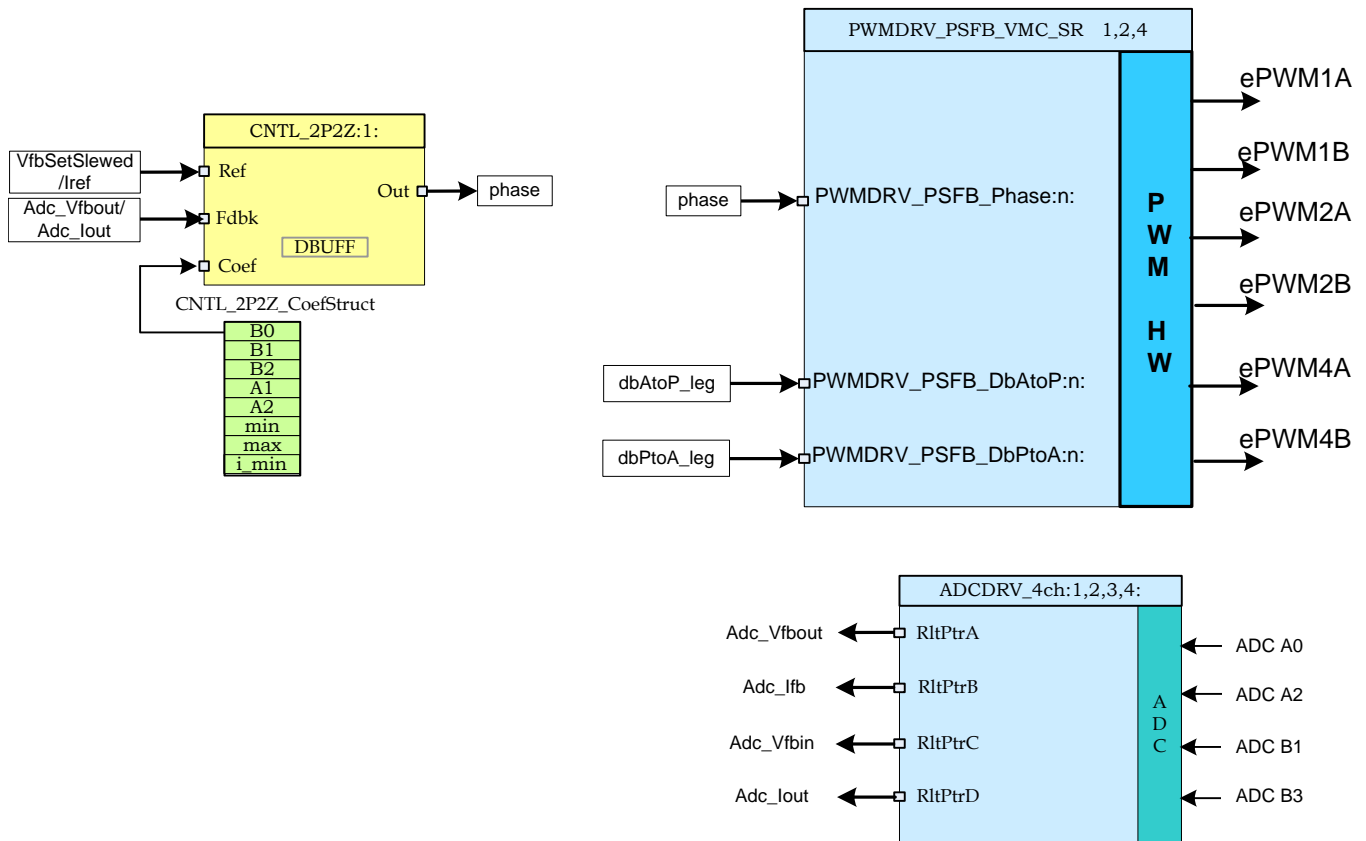


図 39. ソフトウェア・ブロック

図 39では、濃い青色のブロックがC2000マイコン上のハードウェア・モジュールを表しています。青色のブロックは、これらのモジュール用のソフトウェア・ドライバです。黄色のブロックは、制御ループ用のコントローラ・ブロックです。ここでは2極/2ゼロのコントローラを使用していますが、PI/PIDや3極/3ゼロなど、このアプリケーションに実装可能な他の種類のコントローラも使用できます。図 40に示すように、このようなモジュール型のライブラリ構造によって、システム全体のソフトウェアの流れが見やすく、理解しやすくなっています。この構造により、各種の機能の使用、追加、削除が簡単になります。このプロジェクトでは、段階的ビルドのアプローチを実装することで、これらのメリットを示しており、これについては 5.2でさらに詳しく説明します。

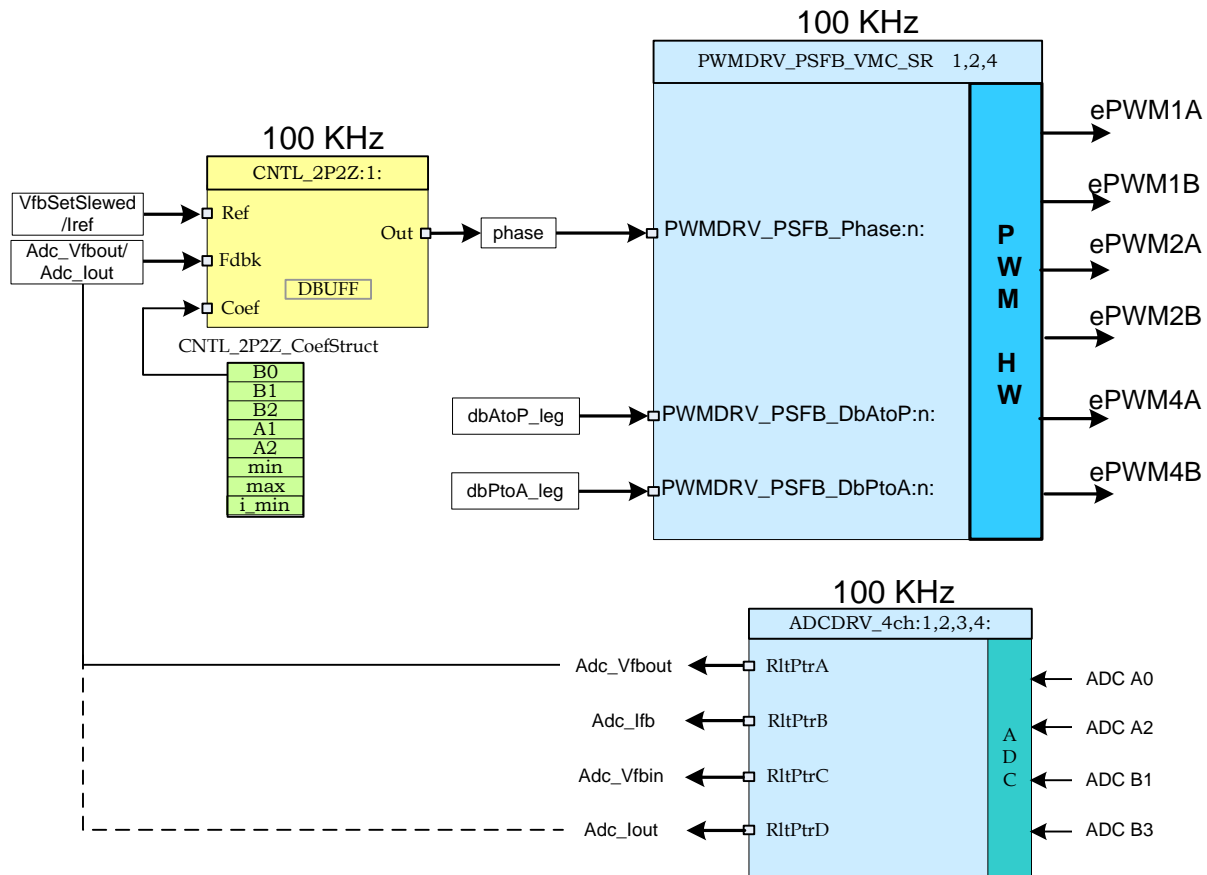


図 40. 制御フロー

システムは、1つの電圧/電流帰還ループによって制御されます。図 40には、制御ブロックの実行レートも示されています。たとえば、電圧コントローラは100kHzのレート(PWMスイッチング周波数と同じ)で実行されます。図 40で実装されている制御について次に説明します。

センスされた出力電圧または電流(Adc_Vfbout/Adc_lout)は、VMCモードでは電圧リファレンス・コマンド(Vref)のスルー・バージョン(VfbSetSlewed)と比較され、ACMCモードでは電流リファレンス・コマンド(Iref)と比較されます。コントローラの出力によって、フルブリッジの2つのレグを駆動するPWM信号間の位相シフトを直接制御します。それによって、出力電圧のレギュレーションに使用されるフルブリッジの2つのレグ間の位相オーバーラップの大きさが決まります。dbAtoP_legおよびdbPtoA_legの値は、それぞれフルブリッジのアクティブ・パッシブ・レグおよびパッシブ・アクティブ・レグに対するデッドバンドの値を決定します。これらの値を使用して、負荷範囲全体にわたるZVS/LVSを実現します。

5.2 段階的ビルド

このプロジェクトは、2つの段階的ビルドに分かれています。このアプローチによって、基板およびソフトウェアについて理解し習熟しやすくなります。また、このアプローチは、基板のデバッグやテストにも適しています。表 3に、ビルド・オプションを示します。特定のビルド・オプションを選択するには、表 3に示されるように、Bi-dir-Settings.hファイルでマクロINCR_BUILDを対応するビルド選択に設定します。ビルド・オプションを選択したら、rebuild-allコンパイラ・オプションを選択して、プロジェクト全体をコンパイルします。ビルド・オプションの実行については、6で詳しく説明します。

表 3. 降圧モードの段階的ビルド・オプション

オプション	説明
INCR_BUILD = 1	ADC帰還を使用した開ループPSFB駆動 (PWM駆動回路およびセンシング回路をチェック)
INCR_BUILD = 2	閉ループ (VMC/ACMCモードでのフルPSFB)

6 降圧モード – 段階的ビルドの実行手順

メイン・ソース・ファイル、ISRアセンブリ・ファイル、およびシステムを起動するCフレームワーク用のプロジェクト・ファイルは、次のディレクトリに格納されています (最新バージョンのソフトウェア・パッケージを使用してください)。

- `..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12lv1_00_00_00\Buck_Mode`

図 7 に示されている 6V~13.5V の DC 電源および 400V の 負荷は、降圧モードの動作には必要ありません。

WARNING

基板上には高電圧が印加されています。ラボ環境で経験を積んだ電源技術者のみが基板を取り扱う必要があります。この基板を安全に評価するには、適切な絶縁型の高電圧 DC 電源を使用してください。基板に DC 電源を印加する前に、電圧計および適切な抵抗性負荷または電子負荷を出力に接続する必要があります。電源が印加されている間は、基板に触れないでください。

Buck_Mode ソフトウェア内のサンプルをビルドして実行する方法の手順を以下に示します。

6.1 ビルド1: ADC帰還を使用した開ループ・チェック

目的

このビルドの目的は、システムの開ループ動作を評価し、PWM および ADC ドライバ・モジュールを検証し、基板上の MOSFET ドライバ回路とセンシング回路を検証し、Code Composer Studio™ (CCS) の動作に習熟することです。このシステムは開ループで動作しているため、ADC での測定値はこのビルドでの計測にのみ使用されます。このセクションでは、プロジェクトのビルドと実行に必要な手順について説明します。

概要

ビルド1のソフトウェアは、位相シフト・フルブリッジPWMドライバ・モジュールをすばやく評価できるように構成されています。これは、オシロスコープで出力波形を表示し、CCS上で位相を対話的に調整しながら、出力電圧に対する位相変化の影響を観測することで行います。ADCドライバ・モジュールの評価は、ADCでサンプリングされたデータを監視ビューで表示することによって行います。

PWMおよびADCドライバのマクロのインスタンス化は、_DPL_ISRの内部で実行されます。図 41に、このビルドの各ブロックを示します。ePWM1AおよびePWM1BはそれぞれQ2およびQ3フルブリッジ・スイッチを駆動し、ePWM2AおよびePWM2BはそれぞれQ1およびQ4フルブリッジ・スイッチを駆動します。ePWM4AおよびePWM4Bは、それぞれQ6およびQ5同期整流器スイッチを駆動します。

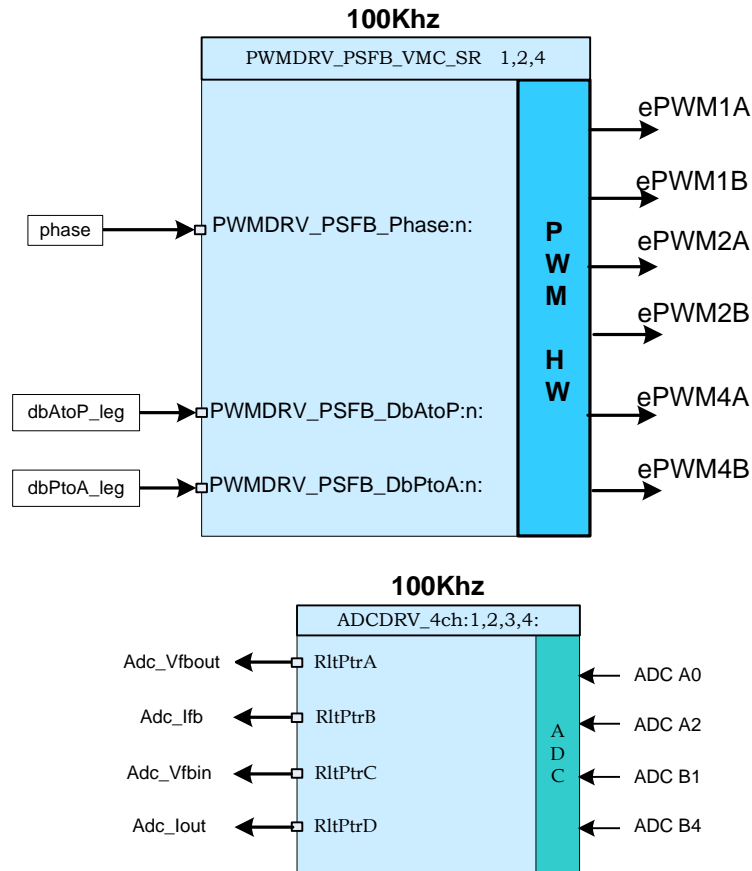


図 41. ビルド1のソフトウェア・ブロック

これらのPWM信号は、100kHzの周波数(つまり、10 μ sの周期)で生成する必要があります。マイコンが60MHzで動作している場合、ePWM1、ePWM2、またはePWM4の時間ベース・カウンタの1カウントは16.667nsに対応します。ePWM1、ePWM2、またはePWM4の時間ベース・カウンタの1カウントが16.667nsに対応するということは、10 μ sのPWM周期が時間ベース・カウンタ(TBCNT1、TBCNT2、およびTBCNT4)の600カウントに相当することを意味します。ePWM1およびePWM2モジュールはアップカウント・モードで動作するよう構成されている一方、ePWM4はアップダウン・カウント・モードで動作します。ePWM1AとePWM1Bの出力は50%のデューティ・サイクルで動作し、相補的になっています。同様に、ePWM2AとePWM2Bの出力も50%のデューティ・サイクルで動作し、相補的になっています。ePWM2の時間ベース・カウンタの位相は、ePWM1の位相を基準にして動的に変化させることができます。図 37に、これらのPWM波形を示します。

PWMドライバ・モジュールへの位相入力によって、PWM1およびPWM2時間ベース・カウンタ間の位相シフトの大きさが決まります。この位相値によって、フルブリッジの対角線上に配置されたスイッチ・ペアを駆動するPWM信号間のオーバーラップの大きさが制御されます。位相が増加するとオーバーラップが大きくなり、2次側に転送されるエネルギーの量が増加します。TBPHS2の値は、入力位相コマンドから得られます。

表 4 に、TBPRDの値が599のときのTBPHS2の値の例を示します。

表 4. 位相値

位相 (Q24)	TBPHS2 = (位相 × TBPRD ÷ 2 ²⁵)	位相シフト(度)
2097152d	37	22.5
8388608d	149	90
16776704d	299	180

フルブリッジの対角線上の各スイッチ・ペアは、1回のPWM周期内で1回オーバーラップします。最大のオーバーラップが発生するのは、位相シフトが約180度のときです。ISRアセンブリは、ePWM1のZRO (TBCNT1 = 0) イベントによってトリガされます。このISRで、制御ドライバのマクロが実行され、TBPHS2およびTBPHS4レジスタが更新されます。

ADC入力がどこでサンプリングされるかを考慮すると、ADC入力信号の完全性が重要になります。これは、ADCがアナログ・ドメインとデジタル・ドメイン間のインターフェイスとして機能するためです。電源段でスイッチをオン/オフすると、このときにセンスされる信号にノイズや外乱が生じる場合があります。このノイズがADC入力上に現れないように、これらの信号上にはフィルタが用意されていますが、このような外乱を避けるために、すべてのADC入力を一度にサンプリングしてください。

センシングされた出力電圧は、スイッチング・サイクル内で、出力電圧値がその平均値に近くなる適切な点でサンプリングする必要があります。ADC入力信号を一度にサンプリングすることで、できる限りノイズの少ないサンプルを取得できるほか、平均の出力電圧をサンプリングできます。フルブリッジでは、2つの対向スイッチ間のオーバーラップの midpoint でサンプリングを行います (オーバーラップとは、両方のスイッチが同時にオンになっている時間を指します)。つまり、MOSFETのスイッチングからできる限り離れたタイミングでサンプリングします。MOSFETのスイッチングからできる限り離れたタイミングでサンプリングすることにより、ADC結果にスイッチング・ノイズが反射されるのを避けることができます。C2000デバイスのADCおよびPWMモジュールは柔軟性が高いため、ADC変換を精密かつ柔軟にトリガできます。ADCドライバ・モジュールを使用して12ビットのADC結果を読み取り、それをQ24値に変換します。各PWMサイクル中に、PWM2 SOCA (変換Aの開始) によって5回のADC変換がトリガされます。

保護

このプロジェクトのシャットダウン・メカニズムでは、オンチップのアナログ・コンパレータ1を使用して、トランスの高電圧巻線電流に対する過電流保護を実装しています。基準トリップ・レベルは内部の10ビットDACを使用して設定され、このコンパレータの反転端子に供給されます。コンパレータの出力は、センスされた電流が設定制限値を上回るときにePWM1、ePWM2、およびePWM4にワンショットのトリップ・アクションを生成するよう構成されています。C2000デバイスのトリップ・メカニズムの柔軟性により、異なるトリップ・イベントに対して異なるアクションを実行することが可能になっています。このプロジェクトでは、電源段を保護するために、ePWM1A、ePWM1B、ePWM2A、ePWM2B、ePWM4A、およびePWM4Bの出力が直ちにLowになります。これらの出力は、デバイスのリセットが実行されるまでの間、Lowに保持されます。低電圧および過電圧保護は実装されていませんが、完全な双方向プロジェクト (Bi_Directional_Full) では実装されています。

6.1.1 手順

CCSを起動し、プロジェクトを開く

このビルドをすばやく実行するには、次の手順に従います。

- 2で示したとおりに、基板の上すべてのジャンパが正しく取り付けまたは取り外されていることを確認します。
- F28035 controlCARDを100ピンDIMMコネクタに挿入します。
- 12V DCのベンチ電源をTP10とTP11の間に正しい極性で接続します。
- 400VのプログラマブルDC電源を400V入力コネクタに接続し、12V負荷を12Vコネクタに接続します (この負荷は基板の定格を超えないようにしてください)。

5. PCと基板の間にUSB-B/USB-Aケーブルを接続します。

注: この時点では、どの電源もオンにしないでください。

6. 基板をJTAG接続でテストするのが初めてである場合は、xds100v2-FT_Prog_v2.2.zipファイル内のprogram_ftdi.batファイルを実行して、基板上のFTDIチップをプログラミングします。
7. Code Composer Studio (CCSv5以降)を起動します。
8. Code Composer Studioを画面上で最大化します。
9. Welcome画面が表示された場合は閉じます。

注: プロジェクトには、マイコンのハードウェアで実行できる実行形式出力ファイル(.out)を開発するために必要な、すべてのファイルとビルド・オプションが含まれています。

10. メニュー・バーで、“Project” → “Import Existing CCS/CCE Eclipse Project”の順にクリックします。
11. ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\v1_00_00_00\Buck_Modeディレクトリを選択します。
12. “Projects”タブで、Bi-dir_Buckにチェックマークが付いていることを確認します。
13. “Finish”をクリックします。

注: このプロジェクトは、すべての必要なツール(コンパイラ、アセンブラ、リンカ)を使用してビルドを実行します。

14. 左側の“Project”ウィンドウで、プロジェクトの左側にある“+”をクリックします。図 42を参照してください。

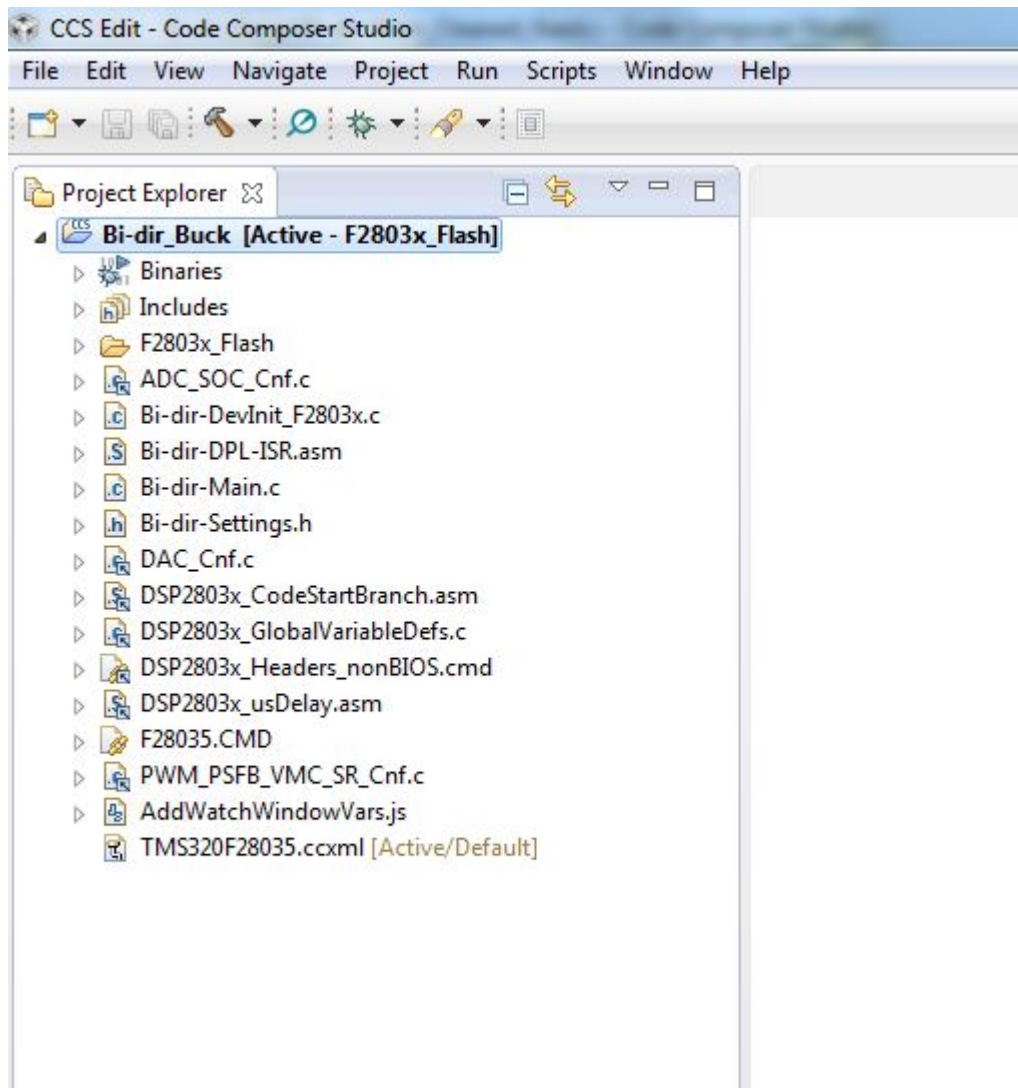


図 42. 降圧モードのCCSプロジェクト・ウィンドウ

15. メイン・ファイル内にあるビルド2の初期化コードを見つけます。
16. コードの内容を確認します。(このコードによって、すべての制御ブロックが設定、初期化され、制御フロー内で接続されます。)

デバイスの初期化、メイン、およびISRファイル

注: ソース・ファイルには一切変更を加えないでください。

1. Bi-dir-DevInit_F2803x.cをダブルクリックします。
2. システム・クロック、ペリフェラル・クロックのプリスケール、およびペリフェラル・クロックのイネーブルが設定されていることを確認します。
3. 共有GPIOピンが設定されていることを確認します。
4. Bi-dir-Main.cをダブルクリックします。
5. DeviceInit()関数の呼び出し、各段階的ビルド・オプションのコード、ISR初期化、およびバックグラウンドfor(;;)ループを表示します。
6. メイン・ファイル内にあるビルド1のコードを見つけます。
7. コードの内容を確認します。

注: このビルド1に固有のコードで、PWMDRV_PSFV_VMC_SRブロックが接続および初期化されます。

8. Bi-dir-DPL-ISR.asmをダブルクリックします。

注: _DPL_Initおよび_DPL_ISRセクションで、それぞれ初期化およびランタイム用のPWMおよびADCドライバ・マクロのインスタンス化が実行されます。確認を終えたファイルは閉じてかまいません。

プロジェクトのビルドとロード

1. Bi-dir-Settings.hファイルで、段階的ビルド・オプションとして1を選択します。

注: 以前に別のオプションをビルドした場合は、プロジェクト名を右クリックして、“Clean Project”をクリックします。

2. “Project” → “Build All”の順にクリックします。
3. 12V DCベンチ電源をオンにします。
4. “Debug”ボタンをクリックします。

注: ビルド1のコードがコンパイルされてロードされます。

F28035ターゲット構成を作成するか、またはF28035.ccxmlファイルから構成をコピーするには、次の手順に従います。

1. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
2. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
3. “Run”→“Load”→“Load Program”の順に選択します。
4. Buck_Mode\F2803x_Flash\Bi-dir_Buck.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

5. main()の先頭でプログラムを停止します。

デバッグ環境ウィンドウ

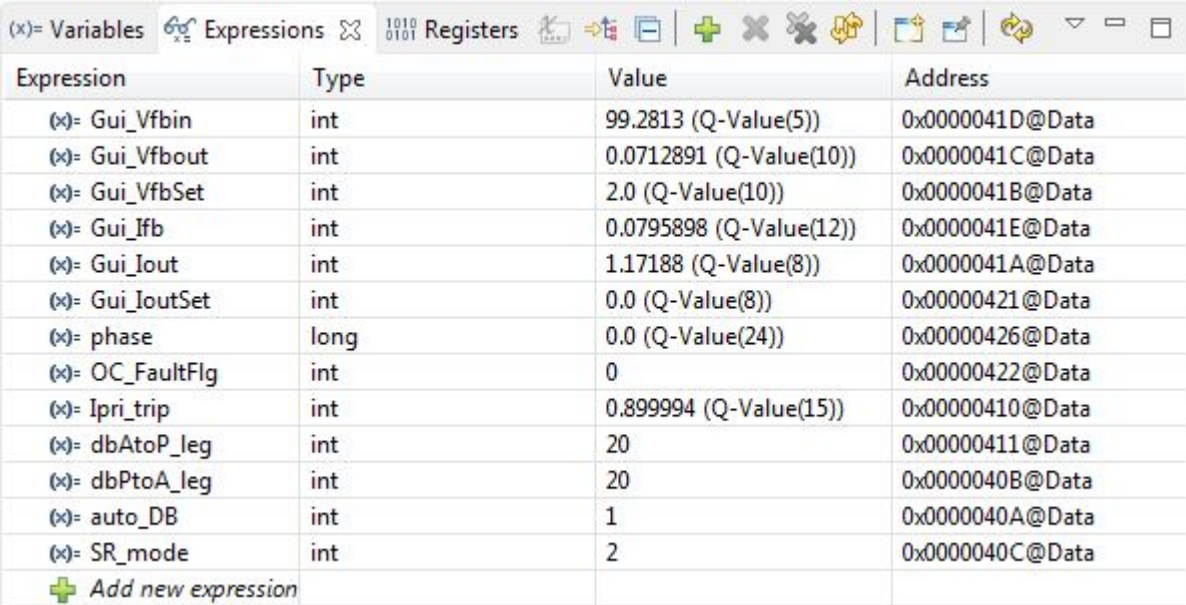
コードのデバッグ中に、ローカル変数およびグローバル変数を監視します。CCSのメモリ・ビューおよび監視ビューを使用できます。CCSでは、時間ドメインと周波数ドメインのプロットを作成できます。波形はグラフ・ウィンドウを使用して表示できます。

1. メニュー・バーで、“View” → “Scripting console”をクリックします。
2. スクリプト・コンソールの“Open File ()”コマンドを使用して、プロジェクト・ディレクトリから AddWatchWindowVars.js ファイルを開きます。

これにより、“Expressions”ウィンドウの各項目が設定されます。“Expressions”ウィンドウの外観については、[図 43](#)を参照してください。

注: いくつかの変数はメイン・コードのこの時点では初期化されておらず、不定の値が格納されています。

OC_FaultFlgがセットされている場合は、過電流状態によってPWM出力がシャットダウンされることを示しています。PWM出力は、デバイスがリセットされるまでこの状態に保持されます。Ipri_trip変数は、オンチップ・コンパレータ1に対する内部10ビットDACの基準レベルを設定します。これはQ15値です。




Expression	Type	Value	Address
(x) Gui_Vfbin	int	99.2813 (Q-Value(5))	0x0000041D@Data
(x) Gui_Vfbout	int	0.0712891 (Q-Value(10))	0x0000041C@Data
(x) Gui_VfbSet	int	2.0 (Q-Value(10))	0x0000041B@Data
(x) Gui_Ifb	int	0.0795898 (Q-Value(12))	0x0000041E@Data
(x) Gui_Iout	int	1.17188 (Q-Value(8))	0x0000041A@Data
(x) Gui_IoutSet	int	0.0 (Q-Value(8))	0x00000421@Data
(x) phase	long	0.0 (Q-Value(24))	0x00000426@Data
(x) OC_FaultFlg	int	0	0x00000422@Data
(x) Ipri_trip	int	0.899994 (Q-Value(15))	0x00000410@Data
(x) dbAtoP_leg	int	20	0x00000411@Data
(x) dbPtoA_leg	int	20	0x0000040B@Data
(x) auto_DB	int	1	0x0000040A@Data
(x) SR_mode	int	2	0x0000040C@Data
+ Add new expression			

図 43. 降圧モードの“Expressions”ウィンドウ: ビルド1

リアルタイム・エミュレーションの使用

リアルタイム・エミュレーションは、マイコンの動作中にCCS内のウィンドウを最大10Hzのレートで更新できるようにする特別なエミュレーション機能です。このエミュレーションにより、グラフや監視ビューが更新され、また、マイコンの動作に影響を与える監視またはメモリ・ウィンドウ内の値を変更することができます。このエミュレーションは、制御パラメータをその場で調整する際に便利です。



リアルタイム・モードをイネーブルにするには、次の手順に従います。

1. 水平ツールバー上のボタンにマウス・ポインタを置きます。
2.  **Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)** をクリックします。

注: メッセージ・ボックスが表示されたら、“YES”を選択して、デバッグ・イベントをイネーブルにします。これにより、ステータス・レジスタ1 (ST1) のビット1 (DGBMビット) が0に設定されます。

DGBMは、デバッグ・イネーブル・マスク・ビットです。DGBMビットが0に設定されているときは、メモリおよびレジスタの値をホスト・プロセッサに渡して、デバッガ・ウィンドウを更新できます。

開いているまたは更新中のウィンドウや変数の数が多すぎるときには、連続的に更新を実行すると、エミュレーション・リンクの帯域幅が限られているために、更新頻度が遅くなる場合があります。“Expressions”ウィンドウの更新レートを変更するには、次の手順に従います。

1. “Expressions”ウィンドウで  を右クリックします。
2. “Continuous Refresh Interval...”を選択します。
3. 連続更新間隔の値を変更します。(通常、これらのテストには、1000msのレートで十分です。)
4. 監視ビューの  をクリックします。

コードの実行

1. ツールバーの“Run”をクリックします。
2. 監視ビューで位相の変数を0.015625 (Q24) に設定します。

注: この変数は、PWMDRV_PSFV_VMC_SRモジュールへの位相シフト・コマンドを規定します。位相に0.005未満の値は使用しないでください。

3. DC出力のPSFBシステムに12V出力で約3A～6Aの電流が流れるように、適切な抵抗性負荷を適用します。

注: TIでは、絶縁型DC電源を使用して、400VのDC入力を基板に供給することを推奨します。

4. J1、J2の入力に390V DCの電源を供給します。
5. 監視ビューで位相を大きな値(0.1など)に設定して、位相コマンドを増加させます。
6. 出力電圧の上昇を観測します。

注: 電圧が基板の定格を超えないようにしてください。

特定の位相値で動作しているときには、負荷を急激に減少させると出力電圧が急上昇する場合があります。ビルド1での動作時には、負荷を急に变化させたり位相コマンドを大幅に増加させたりしないようにしてください。

7. 異なる位相値に対して、監視ビューで異なるADC結果を観測します。

図 44 は、入力電圧が約390V、負荷が12V出力で約6A (2Ω) のときの0.38 (Q24) の位相コマンドに対するシステム動作の監視ビューを示しています。

Expression	Type	Value	Address
(x)- Gui_Vfbin	int	388.25 (Q-Value(5))	0x0000041D@Data
(x)- Gui_Vfbout	int	11.9902 (Q-Value(10))	0x0000041C@Data
(x)- Gui_VfbSet	int	2.0 (Q-Value(10))	0x0000041B@Data
(x)- Gui_Ifb	int	0.468994 (Q-Value(12))	0x0000041E@Data
(x)- Gui_Iout	int	6.27344 (Q-Value(8))	0x0000041A@Data
(x)- Gui_IoutSet	int	0.0 (Q-Value(8))	0x00000421@Data
(x)- phase	long	0.3799999952 (Q-Value(2...))	0x00000426@Data
(x)- OC_FaultFlg	int	0	0x00000422@Data
(x)- Ipri_trip	int	0.899994 (Q-Value(15))	0x00000410@Data
(x)- dbAtoP_leg	int	32	0x00000411@Data
(x)- dbPtoA_leg	int	18	0x0000040B@Data
(x)- auto_DB	int	1	0x0000040A@Data
(x)- SR_mode	int	2	0x0000040C@Data
+ Add new expression			

図 44. 降圧モードの“Expressions”ウィンドウ: ビルド1 (ランタイム)

- 監視ビューからSR_mode変数を0、1、または2に変更して、同期整流器の動作モードを変更します。(デフォルトでは、同期整流器はモード2で動作します。)
- 異なるSRモードで、消費される入力電流の大きさの変化および出力電圧の変化を観測します。
- 同期整流器スイッチを駆動するPWM波形をプローブします。

注: 非常に低い負荷で動作しているときや、出力電圧が非常に低い(6V未満)ときには、SRモードを変更しないでください。そのような場合は、デフォルトのSRモード2を使用してください。


- いくつかの異なる位相値を試します。
- 対応するADC結果を観測します。

注: 位相は小さい間隔で増加させます。常に出力電圧を慎重に観測してください。電圧が基板の定格を超えないようにしてください。

オシロスコープを使用して、PWMゲート駆動信号、入力電圧、電流、出力電圧などの波形をプローブできます。

この絶縁型DC-DCコンバータに対してこれらの高電圧および高電流をプローブする際には、安全性に関する適切な予防措置を実施し、適切な接地要件を考慮してください。

リアルタイム・モードでマイコンを完全に停止させるには、次の手順に従います。

- 400V DC入力をオフにし、数秒間待ちます。
- ツールバーの“Halt”をクリックして、プロセッサを停止します。
-  をクリックして、マイコンのリアルタイム・モードを終了します。
- マイコンをリセットします。

6.2 ビルド2: 閉電圧ループ(フルPSFB)

目的

このビルドの目的は、CCS環境から完全なPSFBプロジェクトの動作を検証することです。

概要

図 45 に、このビルドの各ソフトウェア・ブロックを示します。PWMおよびADCドライバ・ブロックは、ビルド1の場合と同様に使用されます。電圧/電流ループには、2極/2ゼロのコントローラを使用します。アプリケーションの制御ループ要件に応じて、PI、3極/3ゼロなど、他のコントローラ・ブロックも使用できます。図 45 に示されるとおり、電圧ループ・ブロックは100kHzで実行されます。CNTL2P2Zは、IIRフィルタ構造から実現される2次補償回路です。この機能は、すべてのペリフェラルから独立しており、CNF関数の呼び出しを必要としません。

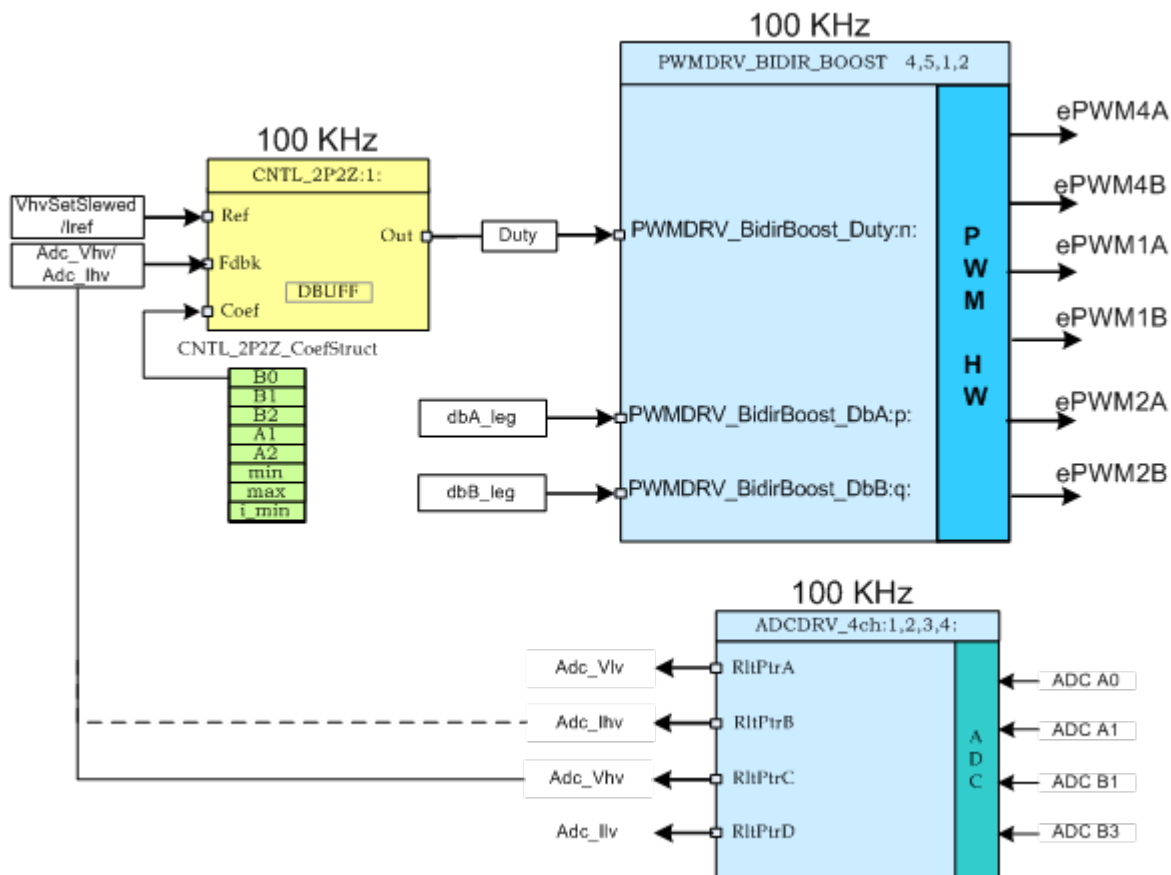


図 45. ビルド2のソフトウェア・ブロック

変更する5つの係数は、構造体CNTL_2P2Z_CoeffStruct1の要素として格納されています。この構造体の他の要素によって、コントローラの出力がクランプされます。システムで複数のループが必要な場合、CNTL_2P2Zブロックは複数回インスタンス化することができます。各インスタンスには、それぞれ個別の係数のセットを使用できます。試行錯誤を通して5つの係数を直接独立に操作することはほぼ不可能であり、数学的な分析や、MATLAB®、Mathcad®などの支援ツールが必要となります。これらのツールは、ボード・プロットや根軌跡など、位相マージンやゲイン・マージンを決定するための機能を提供します。

ループ調整をシンプルに保ち、複雑な計算や分析ツールを不要にするために、TIでは、B0、B1、B2、A1、A2に対して、より直観的な係数ゲインP、I、Dをマッピングすることで、係数選択問題の自由度を5から3へと減らしました。この自由度の減少により、P、I、Dがそれぞれ独立して段階的に調整されます。これらのマッピングの式を以下に示します。

補償回路ブロック (CNTL_2P2Z) は2つの極と2つのゼロを持ち、一般的なIIRフィルタ構造に基づいています。このブロックには、1つのリファレンス入力と1つの帰還入力があります。帰還は、VMCモードの選択時はセンスされた出力電圧 (Adc_Vfbout) であり、ACMCモードの選択時はセンスされた出力電流 (Adc_lout) です。この選択は、Bi-dir-Settings.hファイルから実行できます。コントローラへのリファレンス入力は、モード選択に基づいて、出力電圧リファレンス・コマンド (Vref) のスルー・バージョン (VfbSetSlewed)、またはリファレンス出力インダクタ電流コマンド (Iref) です。伝達関数は式 1 で与えられます。

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (1)$$

PIDコントローラの再帰形式は式 2 で与えられます。

$$u(k) = u(k-1) + b_0 e(k) + b_1 e(k-1) + b_2 e(k-2) \quad (2)$$

ここで、式 3、式 4、式 5 が成り立ちます。

$$b_0 = K_p' + K_i' + K_d' \quad (3)$$

$$b_1 = -K_p' + K_i' - 2 K_d' \quad (4)$$

$$b_2 = K_d' \quad (5)$$

また、Zドメイン伝達関数形式は式 6 のようになります。

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}} = \frac{b_0 z^{-2} + b_1 z + b_2}{z^2 - z} \quad (6)$$

これを一般的な形式と比較すると、PIDはCNTL_2P2Z制御の特殊なケースであり、式 7 が成り立つ場合です。

$$a_1 = -1 \text{ and } a_2 = 0 \quad (7)$$

これらのP、I、およびD係数は、Pgain、Igain、およびDgainです。これらのP、I、およびD係数はQ26形式で使用されます。CCS監視ビューからの調整を単純化するために、これらの3つの係数をさらに0~999の値へと変換します (Pgain_Gui、Igain_Gui、Dgain_Gui)。また、ループ・パラメータも、外部の数学ツール (MATLAB、Mathcadなど) から得られる調整済みの値に基づいて、直接変更できます。ループ係数は、I5Q10形式の変数b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiを使用して直接変更でき、その後で、2P2Zコントローラ用に5個のQ26係数へと変換されます。

このプロジェクトでは、実行中に係数を簡単に切り替えて、両方のループ調整方法を簡単に評価できます。この切り替えは、単にGUI上で2P2Z(On)/PID(Off)ボタンをクリックするか、またはCCSから監視ビュー上でpid2p2z_GUI変数を0または1に変更することによって行えます。このプロジェクトでは、GUI環境からのPIDベースのループ調整 (pid2p2z_GUI = 0) を使用しています。pid2p2z_GUIを1に変更する前に、b2_Gui、b1_Gui、b0_Gui、a2_Gui、およびa1_Gui変数に信頼できる係数値がプログラミングされていることを確認してください。

注: このプロジェクトには、有効なb2_Gui、b1_Gui、b0_Gui、a2_Gui、およびa1_Guiパラメータ値が含まれていません。TIでは、pid2p2z_GUI = 0に保持することで、デフォルトのP、I、Dベースのループを使用することを推奨します。

6.2.1 手順

プロジェクトのビルドとロード

構成済みの作業環境を使用してこのビルドをすばやく実行するには、次の手順に従います。

- 2で示したとおりに、基板上のすべてのジャンパが正しく取り付けまたは取り外されていることを確認します。
- F28035 controlCARDを100ピンDIMMコネクタに挿入します。
- 12V DCのベンチ電源をTP10とTP11の間に正しい極性で接続します。
- 400VのプログラマブルDC電源を400V入力コネクタに接続し、12V負荷を12Vコネクタに接続します (この負荷

は基板の定格を超えないようにしてください。

5. PCと基板の間にUSB-B/USB-Aケーブルを接続します。

注: どの電源もオンにしないでください。

6. 基板をJTAG接続でテストするのが初めてである場合は、xds100v2-FT_Prog_v2.2.zipファイル内のprogram_ftdi.batファイルを実行して、基板上のFTDIチップをプログラミングします。
7. Code Composer Studio (CCSv5以降)を起動します。
8. CCSを画面上で最大化します。
9. Welcome画面が表示された場合は閉じます。

注: プロジェクトには、マイコンのハードウェアで実行できる実行形式出力ファイル(.out)を開発するために必要な、すべてのファイルとビルド・オプションが含まれています。

10. メニュー・バーで、“Project” → “Import Existing CCS/CCE Eclipse Project”の順にクリックします。
11. ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\1v1_00_00_00\Buck_Modeディレクトリを選択します。
12. “Projects”タブで、Bi-dir_Buckにチェックマークが付いていることを確認します。
13. “Finish”をクリックします。

注: このプロジェクトは、すべての必要なツール(コンパイラ、アセンブラ、リンカ)を使用してビルドを実行します。

14. 左側の“Project”ウィンドウで、プロジェクトの左側にある“+”をクリックします。
15. メイン・ファイル内にあるビルド2の初期化コードを見つけます。
16. コードの内容を確認します。(このコードによって、すべての制御ブロックが設定、初期化され、制御フロー内で接続されます。)
17. Bi-dir-Settings.hファイルで、段階的ビルド・オプションとして2を選択します。

注: 以前に別のオプションをビルドした場合は、次の手順に従います。

1. プロジェクト名を右クリックします。
 2. “Clean Project”をクリックします。
 3. “Project” → “Build All”の順にクリックします。
 4. ビルド・ウィンドウでツールが実行されるのを確認します。
-

18. 12V DCベンチ電源をオンにします。
19. “Debug”ボタンをクリックします。

注: ビルド1のコードがコンパイルされてロードされます。

F28035ターゲット構成を作成するか、またはF28035.ccxmlファイルから構成をコピーするには、次の手順に従います。

1. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
2. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
3. “Run”→“Load”→“Load Program”の順に選択します。
4. Buck_Mode\F2803x_Flash\Bi-dir_Buck.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

5. main()の先頭でプログラムを停止します。

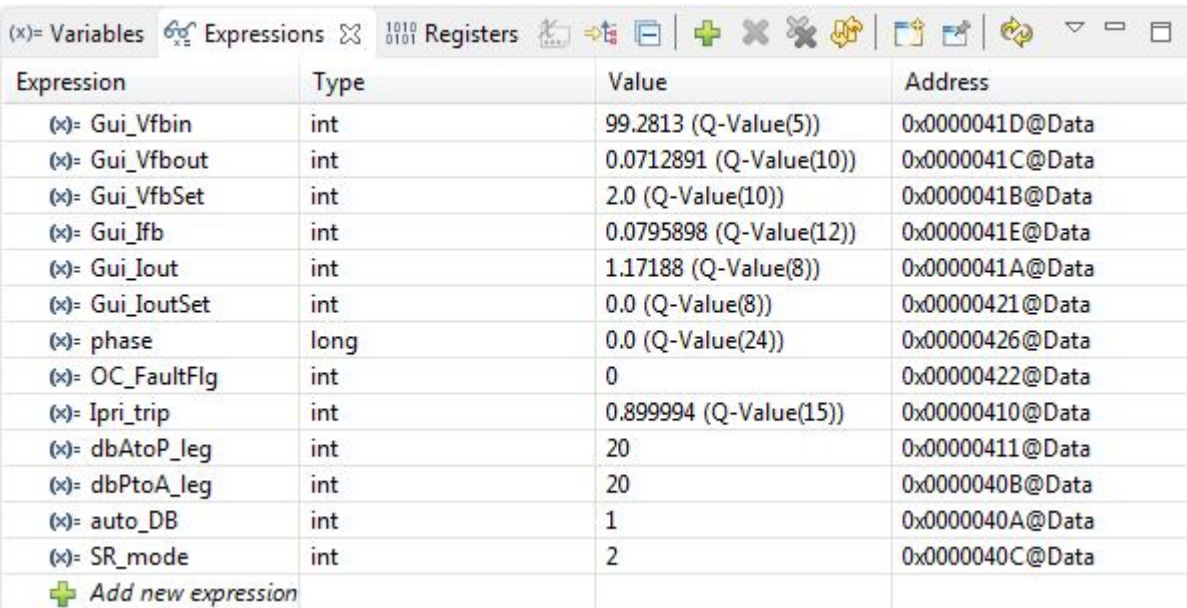
デバッグ環境ウィンドウ

コードのデバッグ中に、ローカル変数およびグローバル変数を監視します。CCSのメモリ・ビューおよび監視ビューを使用できます。CCSでは、時間ドメインと周波数ドメインのプロットを作成できます。波形はグラフ・ウィンドウを使用して表示できます。

1. メニュー・バーで、“View” → “Scripting console”をクリックします。
2. スクリプト・コンソールの“Open File ()”コマンドを使用して、プロジェクト・ディレクトリからAddWatchWindowVars.jsファイルを開きます。

これにより、“Expressions”ウィンドウの各項目が設定されます。“Expressions”ウィンドウの外観については、[図 46](#)を参照してください。

注: いくつかの変数はメイン・コードのこの時点では初期化されておらず、不定の値が格納されています。



Expression	Type	Value	Address
(x)- Gui_VfbIn	int	99.2813 (Q-Value(5))	0x0000041D@Data
(x)- Gui_VfbOut	int	0.0712891 (Q-Value(10))	0x0000041C@Data
(x)- Gui_VfbSet	int	2.0 (Q-Value(10))	0x0000041B@Data
(x)- Gui_Ifb	int	0.0795898 (Q-Value(12))	0x0000041E@Data
(x)- Gui_Iout	int	1.17188 (Q-Value(8))	0x0000041A@Data
(x)- Gui_IoutSet	int	0.0 (Q-Value(8))	0x00000421@Data
(x)- phase	long	0.0 (Q-Value(24))	0x00000426@Data
(x)- OC_FaultFlg	int	0	0x00000422@Data
(x)- Ipri_trip	int	0.899994 (Q-Value(15))	0x00000410@Data
(x)- dbAtoP_leg	int	20	0x00000411@Data
(x)- dbPtoA_leg	int	20	0x0000040B@Data
(x)- auto_DB	int	1	0x0000040A@Data
(x)- SR_mode	int	2	0x0000040C@Data
+ Add new expression			

図 46. 降圧モードの“Expressions”ウィンドウ: ビルド2

注: 監視ビューの追加の変数を確認します。

3. Gui_VfbSetを使用して、出力電圧コマンドを設定します。

コードの実行

1. リアルタイム・モードをイネーブルにします。
2. 監視ビューの連続的更新をイネーブルにし、連続的更新間隔を変更します。
3. ツールバーの“Run”をクリックして、コードを実行します。
4. DC出力のPSFBシステムに12V出力で約3A～6Aの電流が流れるように、適切な抵抗性負荷を適用します。

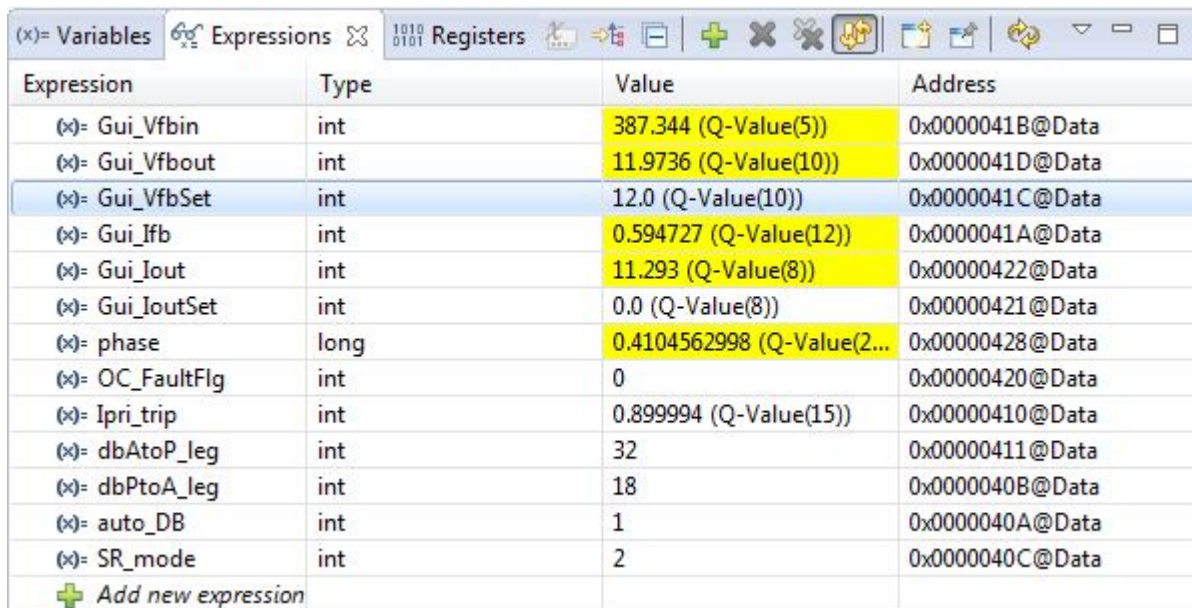
注: TIでは、絶縁型DC電源を使用して、400VのDC入力を基板に供給することを推奨します。

5. 400V入力に390V DC電源を供給します。

6. Gui_VfbSetコマンドを12V (Q10)に変更します。

注: 出力電圧が12Vまで上昇を開始します。この出力電圧の上昇レートは、変数VfbSlewRateの設定によって変更できます。この時点で、Gui_VfbSetコマンドを変更できますが、変更結果が基板の要件を満たすことを確認してください。

図 47 は、入力電圧が約390V、負荷が約12A (1Ω)、12V出力のときのシステム動作に対応する監視ビューを示しています。



Expression	Type	Value	Address
(x)- Gui_VfbIn	int	387.344 (Q-Value(5))	0x0000041B@Data
(x)- Gui_VfbOut	int	11.9736 (Q-Value(10))	0x0000041D@Data
(x)- Gui_VfbSet	int	12.0 (Q-Value(10))	0x0000041C@Data
(x)- Gui_Ifb	int	0.594727 (Q-Value(12))	0x0000041A@Data
(x)- Gui_Iout	int	11.293 (Q-Value(8))	0x00000422@Data
(x)- Gui_IoutSet	int	0.0 (Q-Value(8))	0x00000421@Data
(x)- phase	long	0.4104562998 (Q-Value(2...))	0x00000428@Data
(x)- OC_FaultFlg	int	0	0x00000420@Data
(x)- Ipri_trip	int	0.899994 (Q-Value(15))	0x00000410@Data
(x)- dbAtoP_leg	int	32	0x00000411@Data
(x)- dbPtoA_leg	int	18	0x0000040B@Data
(x)- auto_DB	int	1	0x0000040A@Data
(x)- SR_mode	int	2	0x0000040C@Data
+ Add new expression			

図 47. 降圧モードの“Expressions”ウィンドウ: ビルド2のVMCモード(ランタイム)

- 監視ビューからSR_mode変数を0、1、または2に変更して、同期整流器の動作モードを変更します。(デフォルトでは、同期整流器はモード2で動作します。)
- 異なるSRモードで、消費される入力電流の変化および出力電圧の変化による影響を観測します。(出力電圧または入力電圧への影響はないはずです。)
- 同期整流器スイッチを駆動するPWM波形をプローブします。


注: 非常に低い負荷で動作しているときや、出力電圧が非常に低い(6V未満)ときには、SRモードを変更しないでください。そのような場合は、デフォルトのSRモード2を使用してください。

位相は小さい間隔で増加させます。常に出力電圧を慎重に観測してください。電圧が基板の定格を超えないようにしてください。これらの変化が基板の要件内であることを確認します。

オシロスコープを使用して、PWMゲート駆動信号、入力電圧、電流、出力電圧などの波形をプローブできます。

この絶縁型DC-DCコンバータに対してこれらの高電圧および高電流をプローブする際には、安全性に関する適切な予防措置を実施し、適切な接地要件を考慮してください。

リアルタイム・モードでマイコンを完全に停止させるには、次の手順に従います。

- 400V DC入力をオフにし、数秒間待ちます。
- ツールバーの“Halt”をクリックして、プロセッサを停止します。
-  **Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)** をクリックして、マイコンのリアルタイム・モードを終了します。
- マイコンをリセットします。

コンバータを出力インダクタ制御モード (ACMCモード) で制御するには、次の手順に従います。

1. Bi-dir-Settings.hファイルでVMC0_ACMC1を1に設定します。

注: 以前に別のオプションをビルドした場合は、次の手順に従います。

1. プロジェクト名を右クリックします。
2. “Clean Project”をクリックします。
3. “Project” → “Build All”の順にクリックします。
4. ビルド・ウィンドウでツールが実行されるのを確認します。

2. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
3. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
4. “Run”→“Load”→“Load Program”の順に選択します。
5. Buck_Mode\F2803x_Flash\Bi-dir_Buck.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

6. main()の先頭でプログラムを停止します。
7. リアルタイム・モードをイネーブルにします。
8. 監視ビューの連続的更新をイネーブルにし、連続的更新間隔を変更します。
9. ツールバーの“Run”をクリックして、コードを実行します。
10. DC出力のPSFBシステムに2Ωの抵抗性負荷を適用します。
11. 400V入力に390V DC電源を供給します。
12. Gui_loutSet コマンドを4A (Q12)に変更します。


注: 出力電流は約4A、出力電圧は約8Vになります。一般に、出力インダクタ電流測定の精度は、負荷が大きいほど高くなります。この時点で、Gui_loutSetコマンドを変更できますが、変更結果が基板の要件を満たすことを確認してください。

図 48 に、出力の負荷が2Ωで入力電圧が約390Vのときの、6A電流コマンドに対するシステム動作の監視ビューを示します。

Expression	Type	Value	Address
(x)- Gui_Vfbin	int	387.5 (Q-Value(5))	0x0000041B@Data
(x)- Gui_Vfbout	int	11.3359 (Q-Value(10))	0x0000041D@Data
(x)- Gui_VfbSet	int	2.0 (Q-Value(10))	0x0000041C@Data
(x)- Gui_Ifb	int	0.450684 (Q-Value(12))	0x0000041A@Data
(x)- Gui_Iout	int	6.00391 (Q-Value(8))	0x00000422@Data
(x)- Gui_IoutSet	int	6.0 (Q-Value(8))	0x00000421@Data
(x)- phase	long	0.3539536595 (Q-Value(2...))	0x00000428@Data
(x)- OC_FaultFlg	int	0	0x00000420@Data
(x)- Ipri_trip	int	0.899994 (Q-Value(15))	0x00000410@Data
(x)- dbAtoP_leg	int	32	0x00000411@Data
(x)- dbPtoA_leg	int	18	0x0000040B@Data
(x)- auto_DB	int	1	0x0000040A@Data
(x)- SR_mode	int	2	0x0000040C@Data
+ Add new expression			

図 48. 降圧モードの“Expressions”ウィンドウ: ビルド2のACMCモード(ランタイム)

リアルタイム・モードでマイコンを完全に停止させるには、次の手順に従います。

1. 400V DC入力をオフにし、数秒間待ちます。
2. ツールバーの“Halt”をクリックして、プロセッサを停止します。
3.  をクリックして、マイコンのリアルタイム・モードを終了します。
4. マイコンをリセットします。
5. CCSを閉じます。

7 昇圧モード(プッシュプル) – 機能説明

7.1 電圧モード制御

図 49に示すように、昇圧モードでのVMCモードの実装は、降圧モードでのVMCモードの実装と同様に行われます。この動作モードでの主な違いは、位相シフト制御される降圧モードのコンバータとは異なり、コンバータがデューティ制御コンバータとして動作することです。

昇圧モードでは降圧モード出力インダクタを電流源として使用できるため、このトポロジは電流給電のプッシュプル・コンバータとして動作できます。各プッシュプル・スイッチは、50%を超えるデューティ・サイクルのPWM信号で駆動され、それぞれ互いに180度位相がずれています。このデューティ・サイクルは、高電圧側に転送されるエネルギーの量を規定します。コントローラは、2つのプッシュプル・スイッチを駆動するPWM信号のデューティ・サイクルを直接制御してこのエネルギー転送を制御することにより、出力のレギュレーションを実現します。HV側のフルブリッジ・スイッチはオフに保持し、それらのボディ・ダイオードを整流に使用することができます。この実装では、フルブリッジ・スイッチは、昇圧モードでの同期整流に使用します。図 51に、これらの波形を示します。

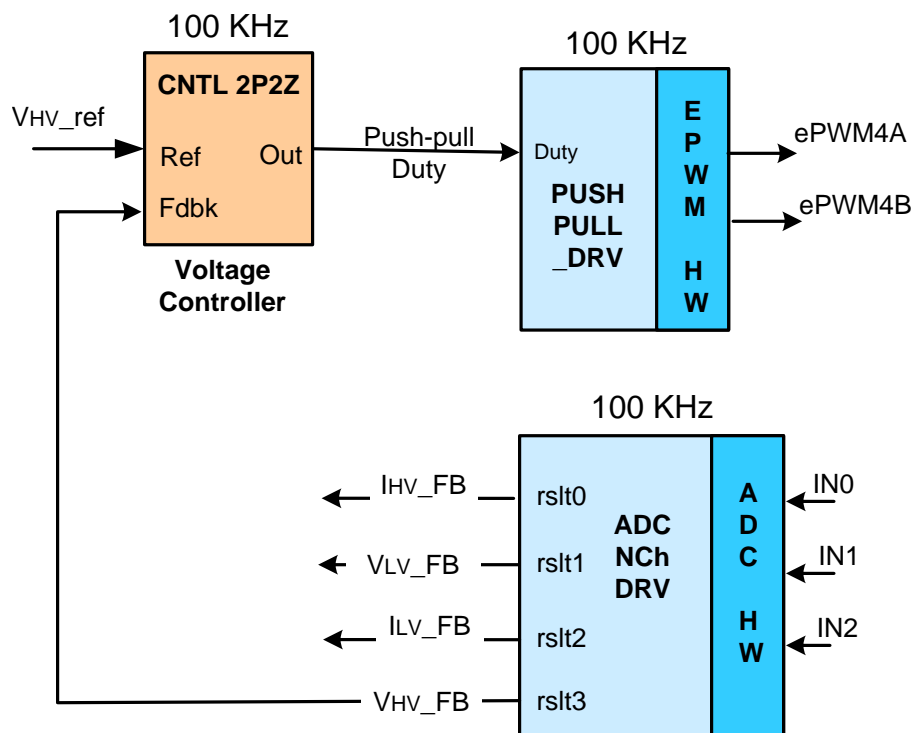


図 49. VMCのブロック図

7.2 出力電流の平均電流モード制御(ACMC)

降圧モードでのACMC実装と同様に、この実装では、1つの平均電流ループを使用して高電圧バッテリーまたはバスの充電電流を制御します(定電流充電)。ACMC実装は、波形生成および電源段制御については7.1のVMC実装と同様です。図50に示すように、コントローラでは、2つのプッシュプル・スイッチを駆動するPWM信号を直接駆動し、そのデューティ・サイクルを制御します。VMCモードと同様に、コントローラはこのデューティ・サイクルを直接制御してエネルギー転送を制御することにより、出力のレギュレーションを行います。

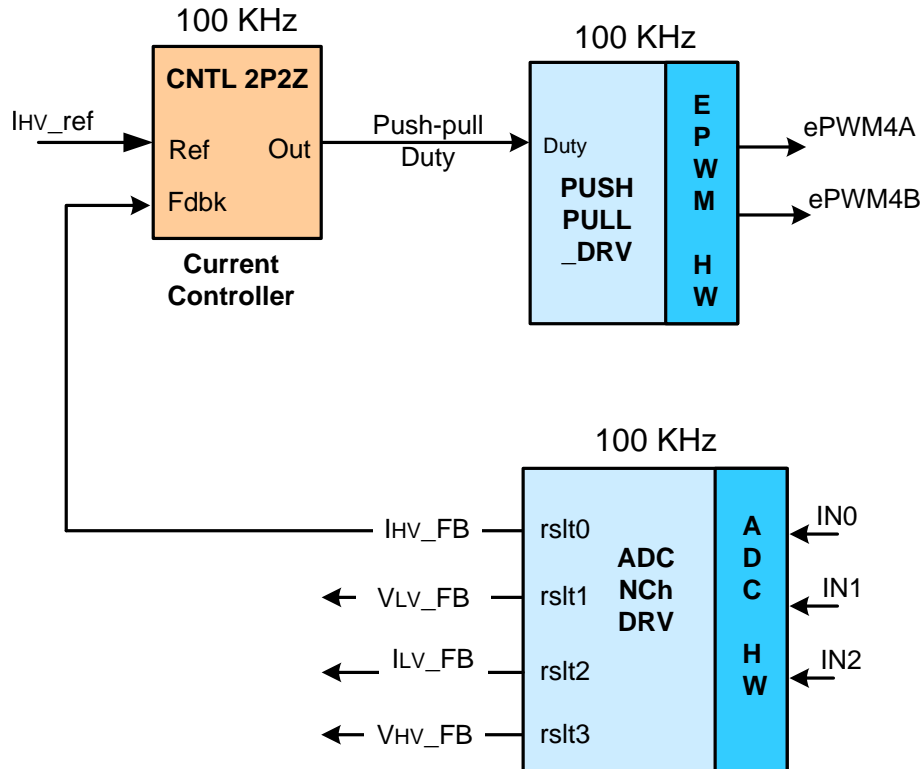


図 50. ACMCのブロック図

図 51 に、昇圧モード動作中のさまざまな波形を示します。

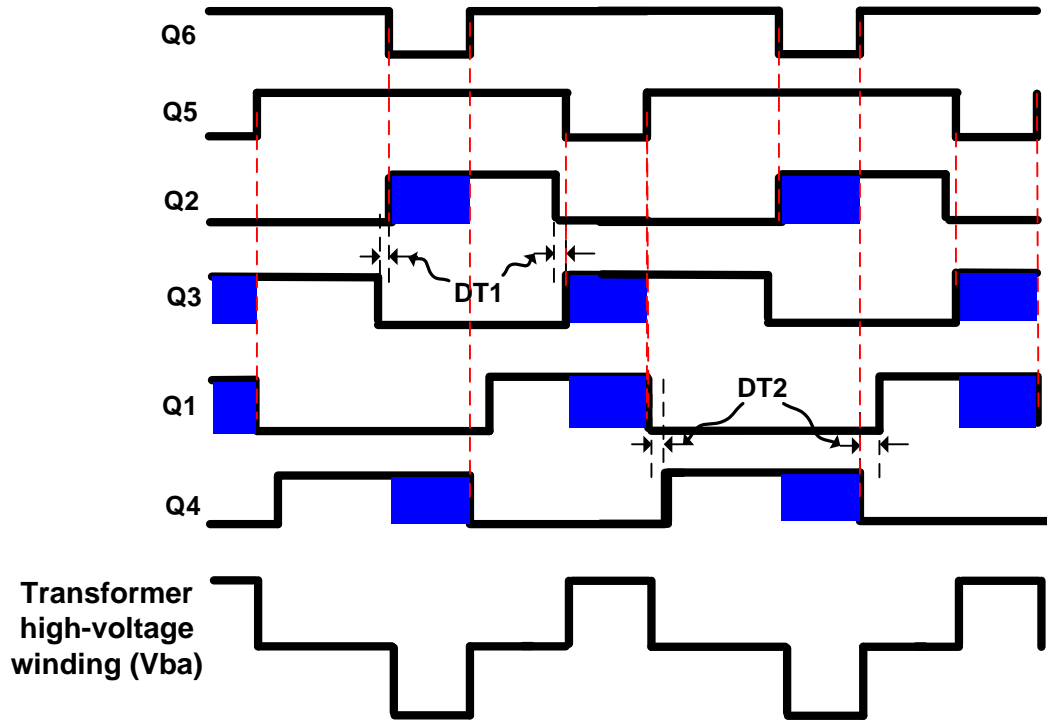


図 51. 昇圧モードの波形

8 昇圧モード – ソフトウェアの概要

8.1 ソフトウェア制御フロー

Bi-dir_Boostプロジェクトでは、Cバックグラウンド/ASM-ISRフレームワークを使用します。このプロジェクトは、アプリケーションのメインのサポート・プログラムとしてCコードを使用し、すべてのシステム管理タスク、意思決定、インテリジェンス、およびホストとのやり取りを実行します。アセンブリ・コードは、すべての重要な制御コードを実行するISRのみに厳密に制限されています。これには一般に、ADCの読み取り、制御用の計算、およびPWMとDACの更新が含まれます。

図 52 に、このプロジェクトの全般的なソフトウェア・フローを示します。

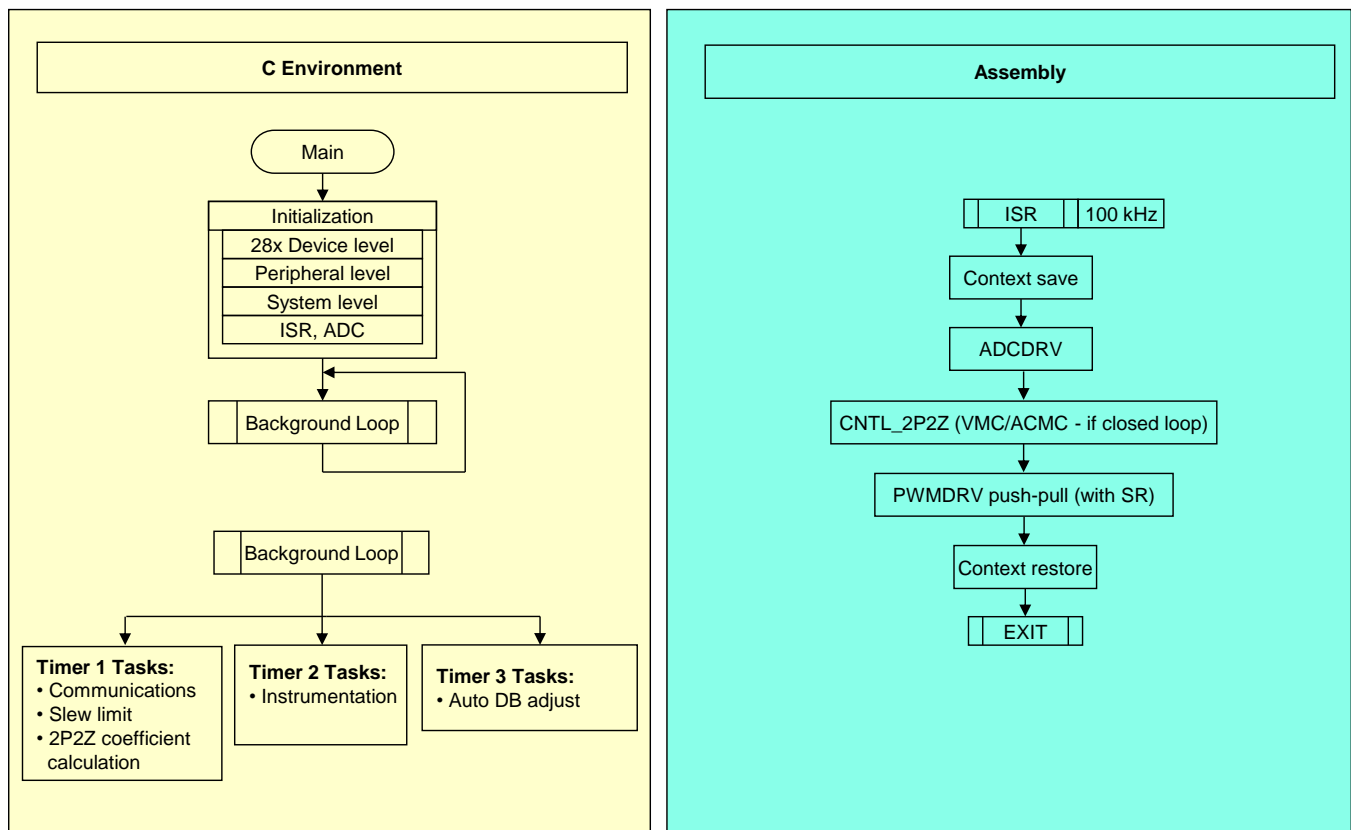


図 52. 昇圧モードのソフトウェア・フロー

このプロジェクトで使用される主要なフレームワークCファイルを以下に示します。

- **Bi-dir-Main.c** – このファイルはアプリケーションを初期化、実行、および管理します。
- **Bi-dir-DevInit_F2803x.c** – このファイルは、マイコンを1回のみ初期化および構成し、クロック、PLL、GPIOの設定などの機能を含みます。

ISRは、以下のファイルから構成されます。

- **Bi-dir-DPL-ISR.asm** – このファイルは、時間が重要となる制御タイプ・コードをすべて含んでいます。このファイルには初期化セクション(1回実行)とランタイム・セクション(PWMスイッチング周波数で実行)が含まれています。

Power Libraryの関数(モジュール)は、このフレームワークから呼び出されます。

ライブラリ・モジュールにはCコンポーネントとアセンブリ・コンポーネントの両方が含まれる場合があります。表 5に、このプロジェクトで使用されるライブラリ・モジュールのCモジュール名および対応するアセンブリ・モジュール名を示します。

表 5. ライブラリ・モジュール

C構成関数	ASM初期化マクロ	ASMランタイム・マクロ
DAC_Cnf.c		
ADC_SOC_Cnf.c	ADCDRV_4CH_INIT m,n,p,q	ADCDRV_4CH m,n,p,q
PWM_PSFV_VMC_SR_Cnf.c	PWMDRV_BIDIR_BOOST_INIT m,n,p,q	PWMDRV_BIDIR_BOOST m,n,p,q
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

図 53 に制御ブロックを示します。

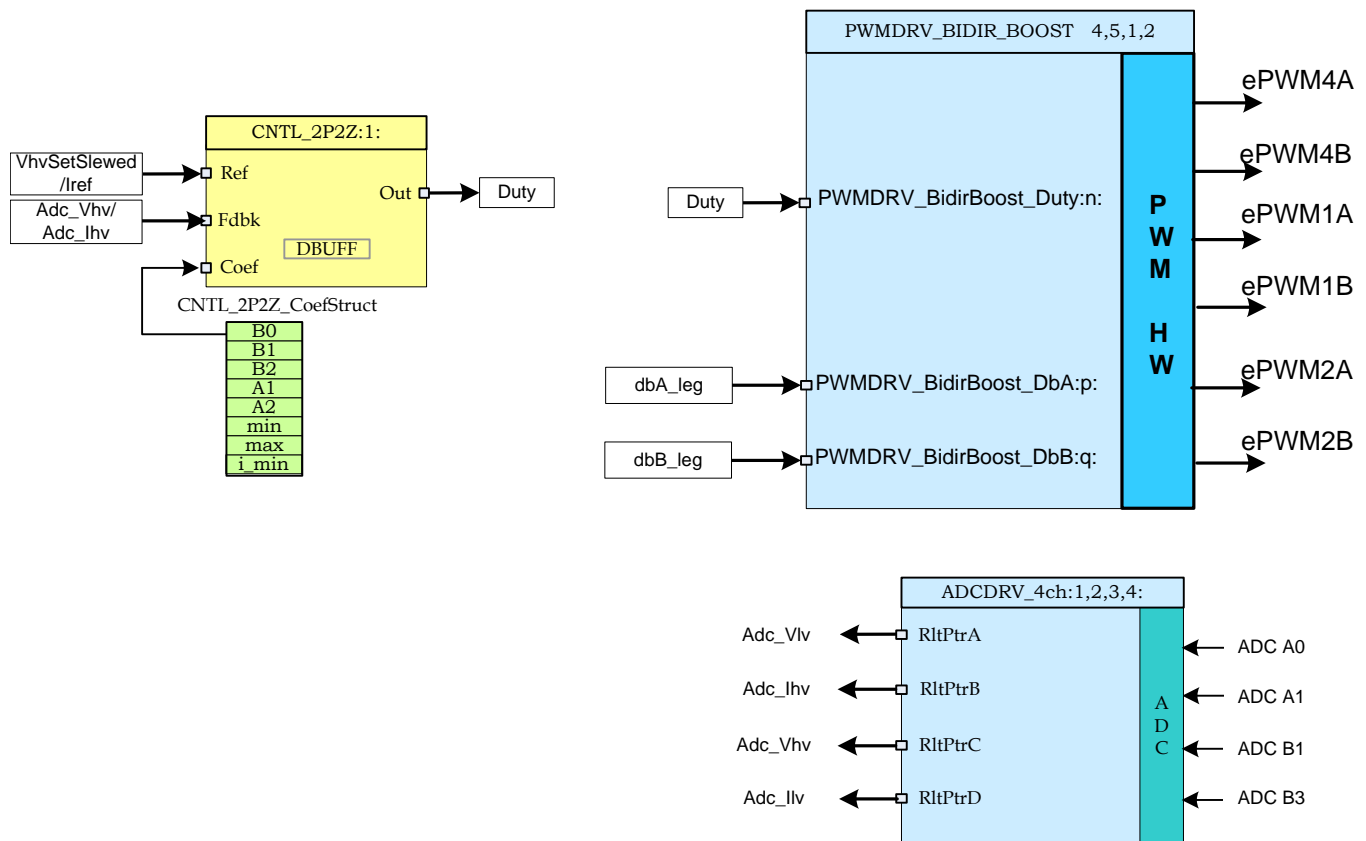


図 53. ソフトウェア・ブロック

図 53では、濃い青色のブロックがC2000マイコン上のハードウェア・モジュールを表しています。青色のブロックは、これらのモジュール用のソフトウェア・ドライバです。黄色のブロックは、制御ループ用のコントローラ・ブロックです。ここでは2極/2ゼロのコントローラを使用していますが、PI/PIDや3極/3ゼロなど、このアプリケーションに実装可能な他の種類のコントローラも使用できます。図 40に示すように、このようなモジュール型のライブラリ構造によって、システム全体のソフトウェアの流れが見やすく、理解しやすくなっています。また、この構造により、各種の機能の使用、追加、削除が簡単になります。このプロジェクトでは、段階的ビルドのアプローチを実装することで、これらのメリットを示しており、これについては次のセクションでさらに詳しく説明します。システムは、1つの電圧/電流帰還ループによって制御されます。

図 54 には、制御ブロックの実行レートも示されています。たとえば、電圧コントローラは100kHzのレート(PWM スイッチング周波数と同じ)で実行されます。

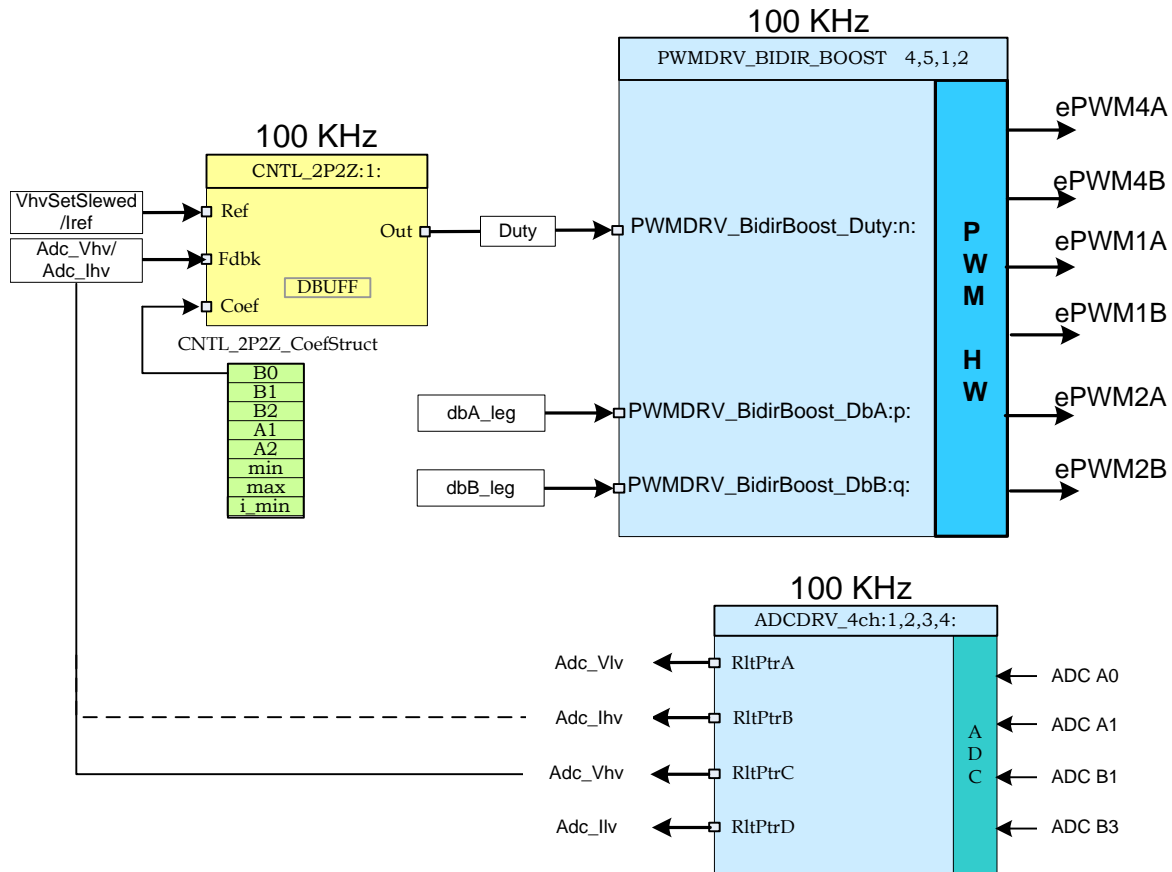


図 54. 制御フロー

センスされた出力電圧/電流 (Adc_Vhv/Adc_lhv) は、電圧/電流コントローラの電圧または電流リファレンス・コマンド (Vref/lref) と比較されます。コントローラ出力によって、2つのプッシュプル・スイッチを駆動するPWM信号のデューティ・サイクルを直接制御します。このデューティ・サイクルは、高電圧側の同期整流に対するフルブリッジの2つのレグ間の位相オーバーラップの大きさを規定します。dbA_legおよびdbB_legの値によって、フルブリッジ・スイッチに適切なターンオン遅延が提供されます。

8.2 段階的ビルド

このプロジェクトは、2つの段階的ビルドに分かれています。これによって、基板およびソフトウェアについて理解し習熟しやすくなります。また、このアプローチは、基板のデバッグやテストにも適しています。表 6に、ビルド・オプションを示します。特定のビルド・オプションを選択するには、表 6に示されるように、Bi-dir-Settings.hファイルにあるマクロINCR_BUILDを対応するビルド選択に設定します。ビルド・オプションを選択したら、rebuild-allコンパイラ・オプションを選択して、プロジェクト全体をコンパイルします。各ビルド・オプションの実行の詳細については、9を参照してください。

表 6. 降圧モードの段階的ビルド・オプション

オプション	説明
INCR_BUILD = 1	ADC帰還を使用した開ループ昇圧駆動 (PWM駆動回路およびセンシング回路をチェック)
INCR_BUILD = 2	開ループ (VMC/ACMCモードでの完全な昇圧モード)

9 昇圧モード – 段階的ビルドの実行手順

メイン・ソース・ファイル、ISRアセンブリ・ファイル、およびシステムを起動するCフレームワーク用のプロジェクト・ファイルは、次のディレクトリに格納されています(最新バージョンのソフトウェア・パッケージを使用してください)。

- `..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\1_00_00_00\Boost_Mode`

図 7 に、昇圧モードのテスト用のハードウェア構成を示します。

WARNING

基板上には高電圧が印加されています。ラボ環境で経験を積んだ電源技術者のみが基板を取り扱う必要があります。この基板を安全に評価するには、適切な絶縁型の高電圧DC電源を使用してください。基板にDC電源を印加する前に、電圧計および適切な抵抗性負荷または電子負荷を出力に接続する必要があります。電源が印加されている間は、基板に触れないでください。

Boost_Modeソフトウェア内のサンプルをビルドして実行する方法の手順を以下に示します。

9.1 ビルド1: ADC帰還を使用した開ループ・チェック

目的

このビルドの目的は、システムの開ループ動作を評価し、PWMおよびADCドライバ・モジュールを検証し、基板上のMOSFETドライバ回路とセンシング回路を検証し、CCSの動作に習熟することです。このシステムは開ループで動作しているため、ADCでの測定値はこのビルドでの計測の目的にのみ使用されます。このセクションでは、プロジェクトのビルドと実行に必要な手順について説明します。

概要

ビルド1のソフトウェアは、電流給電プッシュプルPWMドライバ・モジュールをすばやく評価できるように構成されています。これは、オシロスコープで出力波形を表示し、CCS上でデューティ・コマンドを対話的に調整しながら、出力電圧に対するデューティ・サイクル変化の影響を観測することで行います。ADCドライバ・モジュールの評価は、ADCでサンプリングされたデータを監視ビューで表示することによって行います。

PWMおよびADCドライバのマクロのインスタンス化は、8の_DPL_ISRの内部で実行されます。図 55に、このビルドで使用される各ブロックを示します。ePWM1AおよびePWM1BはそれぞれQ2およびQ3フルブリッジ・スイッチを駆動し、ePWM2AおよびePWM2BはそれぞれQ1およびQ4フルブリッジ・スイッチを駆動します。ePWM4AおよびePWM4Bは、それぞれQ6およびQ5プッシュプル・スイッチを駆動します。

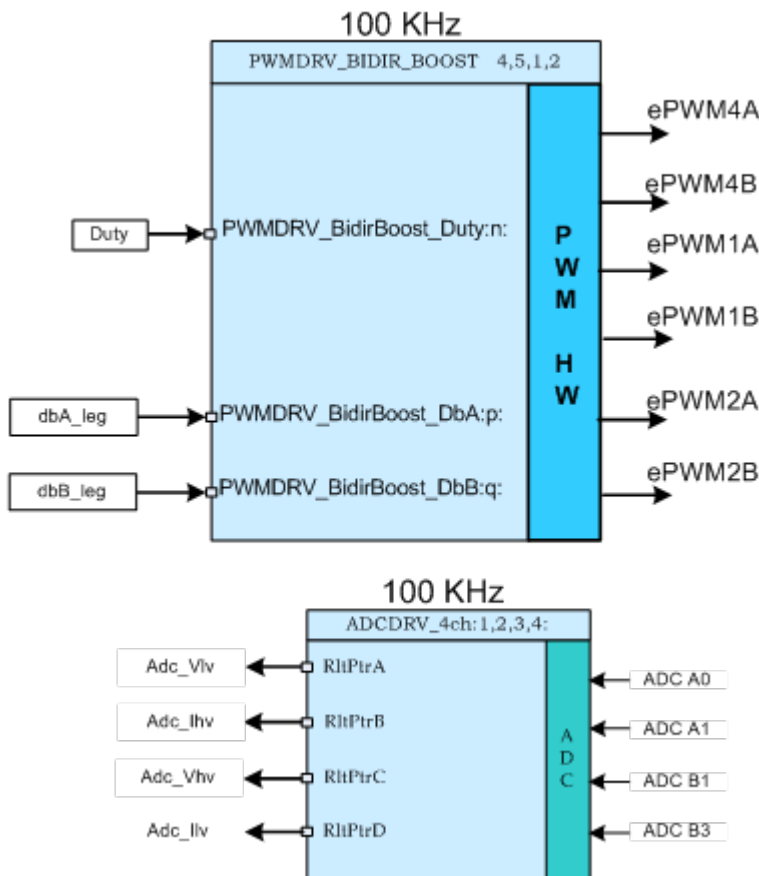


図 55. ビルド1のソフトウェア・ブロック

これらのPWM信号は、100kHzの周波数(つまり、10μsの周期)で生成する必要があります。マイコンが60MHzで動作している場合、ePWM1、ePWM2、またはePWM4の時間ベース・カウンタの1カウントは16.667nsに対応します。ePWM1、ePWM2、またはePWM4の時間ベース・カウンタの1カウントが16.667nsに対応することは、10μsのPWM周期が時間ベース・カウンタ(TBCNT1、TBCNT2、およびTBCNT4)の600カウントに相当することを意味します。ePWM1、ePWM2、およびePWM4モジュールはアップダウン・カウント・モードで動作するよう構成されています。ePWM1AとePWM1Bの出力は50%のデューティ・サイクルで動作し、相補的になっています。同様に、ePWM2AとePWM2Bの出力も50%のデューティ・サイクルで動作し、相補的になっています。ePWM4AおよびePWM4Bは50%を超えるデューティ・サイクルで動作し、互いに位相が180度ずれています。図 51に、これらのPWM波形を示します。

2つのプッシュプル・スイッチを駆動するPWM信号間のオーバーラップが終了すると、対応する対角線上のフルブリッジ・スイッチ・ペアがオンになります。Q6がオフの間は、Q2とQ4を駆動するPWM信号がオーバーラップします。Q5がオフの間は、Q1とQ3を駆動するPWM信号がオーバーラップします。

アセンブリISRは、ePWM1のZRO(TBCNT1 = 0)イベントによってトリガされます。このISRで、制御ドライバのマクロが実行され、プッシュプルのデューティ・コマンドが更新されます。ADCドライバ・モジュールを使用して12ビットのADC結果を読み取り、それをQ24値に変換します。各PWMサイクル中に、PWM2 SOCA(変換Aの開始)によって5回のADC変換がトリガされます。

保護

このプロジェクトで使用されるシャットダウン・メカニズムでは、オンチップのアナログ・コンパレータ1を使用して、トランスの高電圧巻線電流に対する過電流保護を実装しています。LVインダクタ電流の過電流保護は、オンチップ・アナログ・コンパレータ2を使用して実装されます。基準トリップ・レベルは内部の10ビットDACを使用して設定され、これらのコンパレータの反転端子に供給されます。コンパレータの出力は、センスされた電流が設定制限値を上回るときにePWM1、ePWM2、およびePWM4にワンショットのトリップ・アクションを生成するよう構成されています。LV側とHV側両方の過電圧保護は、低速ステートマシン・タスクA1内部のソフトウェアに実装されています。過電圧状態が検出されると、ePWM1、ePWM2、およびePWM4でワンショット・トリップ・アクションが開始されます。

C2000デバイスのトリップ・メカニズムの柔軟性により、異なるトリップ・イベントに対して異なるアクションを実行することが可能になっています。このプロジェクトでは、電源段を保護するために、ePWM1A、ePWM1B、ePWM2A、ePWM2B、ePWM4A、およびePWM4Bが直ちにLowになります。これらの出力は、デバイスのリセットが実行されるまでの間、Lowに保持されます。

9.1.1 手順

CCSを起動し、プロジェクトを開く

このビルドをすばやく実行するには、次の手順に従います。

1. 2で示したとおりに、基板上のすべてのジャンパが正しく取り付けまたは取り外されていることを確認します。
2. F28035 controlCARDを100ピンDIMMコネクタに挿入します。
3. 12V DCのベンチ電源をTP10とTP11の間に正しい極性で接続します。
4. 400VのプログラマブルDC電源を400V入力コネクタに接続し、12V負荷を12Vコネクタに接続します(この負荷は基板の定格を超えないようにしてください)。
5. PCと基板の間にUSB-B/USB-Aケーブルを接続します。

注: この時点では、どの電源もオンにしないでください。

6. 基板をJTAG接続でテストするのが初めてである場合は、xds100v2-FT_Prog_v2.2.zipファイル内のprogram_ftdi.batファイルを実行して、基板上のFTDIチップをプログラミングします。
7. Code Composer Studio (CCSv5以降)を起動します。
8. CCSを画面上で最大化します。
9. Welcome画面が表示された場合は閉じます。

注: プロジェクトには、マイコンのハードウェアで実行できる実行形式出力ファイル(.out)を開発するために必要な、すべてのファイルとビルド・オプションが含まれています。

10. メニュー・バーで、“Project” → “Import Existing CCS/CCE Eclipse Project”の順にクリックします。
11. ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\v1_00_00_00\Boost_Modeディレクトリを選択します。
12. “Projects”タブで、Bi_dir_Boostにチェックマークが付いていることを確認します。
13. “Finish”をクリックします。

注: このプロジェクトは、すべての必要なツール(コンパイラ、アセンブラ、リンカ)を使用してビルドを実行します。

14. 左側の“Project”ウィンドウで、プロジェクトの左側にある“+”をクリックします。図 56を参照してください。

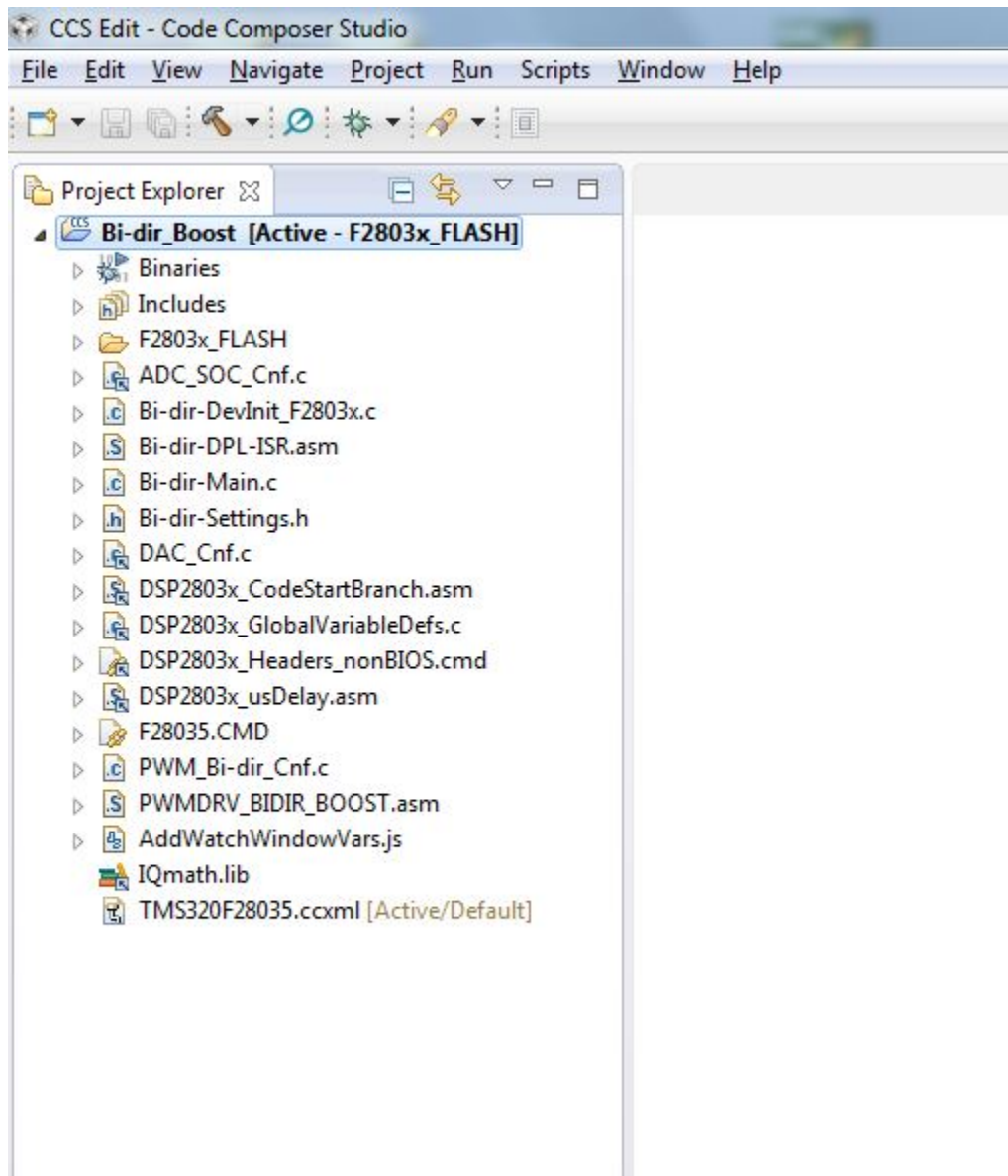


図 56. 昇圧モードのCCSプロジェクト・ウィンドウ

デバイスの初期化、メイン、およびISRファイル

注: ソース・ファイルには一切変更を加えないでください。

1. Bi-dir-DevInit_F2803x.cをダブルクリックします。
2. システム・クロック、ペリフェラル・クロックのプリスケール、およびペリフェラル・クロックのイネーブルが設定されていることを確認します。
3. 共有GPIOピンが設定されていることを確認します。
4. Bi-dir-Main.cをダブルクリックします。
5. DeviceInit()関数の呼び出し、各段階的ビルド・オプションのコード、ISR初期化、およびバックグラウンドfor(;;)ループを表示します。
6. メイン・ファイル内にあるビルド1のコードを見つけます。

7. コードの内容を確認します。

注: これは、制御フロー内でADCDRV_4CHブロックを接続し、初期化するコードです。

8. Bi-dir-DPL-ISR.asmをダブルクリックします。

注: 初期化およびランタイム用のPWMおよびADCドライバ・マクロのインスタンス化は、それぞれ_DPL_Initおよび_DPL_ISRセクションで実行されます。確認を終えたファイルは閉じてかまいません。

プロジェクトのビルドとロード

1. Bi-dir-Settings.hファイルで、段階的ビルド・オプションとして1を選択します。

注: 以前に別のオプションをビルドした場合は、プロジェクト名を右クリックして、“Clean Project”をクリックします。

2. “Project” → “Build All”の順にクリックします。
3. 12V DCベンチ電源をオンにします。
4. “Debug”ボタンをクリックします。

注: ビルド1のコードがコンパイルされてロードされます。

F28035ターゲット構成を作成するか、またはF28035.ccxmlファイルから構成をコピーするには、次の手順に従います。

1. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
2. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
3. “Run” → “Load” → “Load Program”の順に選択します。
4. Boost_Mode\F2803x_Flash\Bi-dir_Boost.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

5. main()の先頭でプログラムを停止します。

デバッグ環境ウィンドウ

コードのデバッグ中に、ローカル変数およびグローバル変数を監視します。CCSのメモリ・ビューおよび監視ビューを使用できます。CCSでは、時間ドメインと周波数ドメインのプロットを作成できます。波形はグラフ・ウィンドウを使用して表示できます。

1. メニュー・バーで、“View” → “Scripting console”をクリックします。
2. スクリプト・コンソールの“Open File ()”コマンドを使用して、プロジェクト・ディレクトリからAddWatchWindowVars.jsファイルを開きます。

これにより、“Expressions”ウィンドウの各項目が設定されます。“Expressions”ウィンドウの外観については、[図 43](#)を参照してください。

いくつかの変数はメイン・コードのこの時点では初期化されておらず、不定の値が格納されています。

OC_FaultFlgがセットされている場合は、過電流状態によってPWM出力がシャットダウンされることを示しています。PWM出力は、デバイスがリセットされるまでこの状態に保持されます。Ihv_trip変数はオンチップ・コンパレータ1に対する内部10ビットDAC基準レベル (HV巻線の過電流スレッショルド)を設定し、Ilv_trip変数はオンチップ・コンパレータ2に対する内部10ビットDAC基準レベル (LVインダクタの過電流スレッショルド)を設定します。これらはQ15値です。を参照してください。

Expression	Type	Value	Address
(x)= Gui_Vhv	int	100.063 (Q-Value(5))	0x00000425@Data
(x)= Gui_VhvSet	int	0.0 (Q-Value(5))	0x0000041C@Data
(x)= Gui_Vlv	int	0.12793 (Q-Value(10))	0x00000430@Data
(x)= Gui_Ihv	int	0.029541 (Q-Value(12))	0x00000428@Data
(x)= Gui_IhvSet	int	0.0 (Q-Value(12))	0x0000041D@Data
(x)= Gui_IlV	int	0.3125 (Q-Value(8))	0x0000042F@Data
(x)= start_converter	int	0	0x0000042D@Data
(x)= Boost_ON	int	0	0x00000429@Data
(x)= BoostDuty	long	5.960464478e-07 (Q-Value(24))	0x0000044A@Data
(x)= OC_FaultFlg	int	1	0x00000407@Data
(x)= HVbus_OV	int	0	0x0000040B@Data
(x)= LVbus_OV	int	0	0x00000406@Data
(x)= Ihv_trip	int	0.949982 (Q-Value(15))	0x00000418@Data
(x)= Ilv_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
(x)= dbA1	int	20	0x00000410@Data
(x)= dbB2	int	20	0x0000040E@Data
(x)= auto_DB	int	1	0x0000040F@Data
+ Add new expression			

図 57. 昇圧モードの“Expressions”ウィンドウ: ビルド1

リアルタイム・エミュレーションの使用

リアルタイム・エミュレーションは、マイコンの動作中にCCS内のウィンドウを最大10Hzのレートで更新できるようにする特別なエミュレーション機能です。このエミュレーションにより、グラフや監視ビューが更新され、また、マイコンの動作に影響を与える監視またはメモリ・ウィンドウ内の値を変更することができます。このエミュレーションは、制御パラメータをその場で調整する際に便利です。

リアルタイム・モードをイネーブルにするには、次の手順に従います。

1. 水平ツールバー上のボタンにマウス・ポインタを置きます。
2. **Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)** をクリックします。


注: メッセージ・ボックスが表示されたら、“YES”を選択して、デバッグ・イベントをイネーブルにします。これにより、ステータス・レジスタ1 (ST1) のビット1 (DGBMビット) が0に設定されます。

DGBMは、デバッグ・イネーブル・マスク・ビットです。DGBMビットが0に設定されているときは、メモリおよびレジスタの値をホスト・プロセッサに渡して、デバッガ・ウィンドウを更新できます。

開いているまたは更新中のウィンドウや変数の数が多すぎる場合には、連続的に更新を実行すると、エミュレーション・リンクの帯域幅が限られているために、更新頻度が遅くなる場合があります。

“Expressions”ウィンドウの更新レートを下げるには、次の手順に従います。

1. “Expressions”ウィンドウで を右クリックします。
2. “Continuous Refresh Interval...”を選択します。

3. 連続更新間隔の値を変更します。(通常、これらのテストには、1000msのレートで十分です。)
4. 監視ビューのをクリックします。

コードの実行

1. ツールバーの“Run”をクリックします。
2. 監視ビューでstart_converter変数が0に設定されていることを確認します。
3. BoostDutyが0.0 (Q24) に設定されていることを確認します。

注: この変数は、PWMDRV_BIDIR_BOOSTモジュールへのデューティ・コマンドを規定します。

4. HV (400V) DC出力に2kΩの抵抗性負荷を適用します。

注: TIでは、絶縁型DC電源を使用して、400V DCおよび12V入力を供給することを推奨します。

5. 400V DC電源を100V DC出力に設定します。
6. この電源をオンにします。
7. start_converterを1に設定します。(400V DCバス電圧が約150Vまで上昇し、400V DC電源電流は0まで減少します。)
8. 監視ビューでBoostDutyを0.1 (Q24) に設定して、デューティ・コマンドを増加させます。
9. 出力電圧が上昇して昇圧動作が行われるのを観測します。

注: 電圧が基板の定格を超えないようにしてください。

特定のBoostDuty値で動作しているときには、負荷を急激に減少させると出力電圧が急上昇する場合があります。ビルド1での動作時には、負荷や入力電圧を急に变化させたりBoostDutyコマンドを大幅に増加させたりしないようにしてください。

10. 異なるBoostDuty値に対して、監視ビューで異なるADC結果を観測します。

図 58 は、LV入力が12V、HV出力の負荷が2kΩのときの、約250VのHV電圧出力を得る0.35 (Q24) の BoostDutyコマンドに対するシステム動作の監視ビューを示しています。(12Vコネクタには100Ωの負荷も接続されています。)

Expression	Type	Value	Address
Gui_Vhv	int	250.188 (Q-Value(5))	0x00000425@Data
Gui_VhvSet	int	0.0 (Q-Value(5))	0x0000041C@Data
Gui_Vlv	int	11.9395 (Q-Value(10))	0x00000430@Data
Gui_Ihv	int	0.39209 (Q-Value(12))	0x00000428@Data
Gui_IhvSet	int	0.0 (Q-Value(12))	0x0000041D@Data
Gui_Ilv	int	4.10938 (Q-Value(8))	0x0000042F@Data
start_converter	int	1	0x0000042D@Data
Boost_ON	int	1	0x00000429@Data
BoostDuty	long	0.3499999642 (Q-Value(24))	0x0000044A@Data
OC_FaultFlg	int	0	0x00000407@Data
HVbus_OV	int	0	0x0000040B@Data
LVbus_OV	int	0	0x00000406@Data
Ihv_trip	int	0.949982 (Q-Value(15))	0x00000418@Data
Ilv_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
dbA1	int	32	0x00000410@Data
dbB2	int	18	0x0000040E@Data
auto_DB	int	1	0x0000040F@Data
+ Add new expression			

図 58. 昇圧モードの“Expressions”ウィンドウ: ビルド1(ランタイム)


11. いくつかの異なるBoostDuty値を試します。
12. ADC結果を観測します。

注: 位相は小さい間隔で増加させます。常に出力電圧を慎重に観測してください。電圧が基板の定格を超えないようにしてください。

オシロスコープを使用して、PWMゲート駆動信号、入力電圧、電流、出力電圧などの波形をプローブできます。

この絶縁型DC-DCコンバータに対してこれらの高電圧および高電流をプローブする際には、安全性に関する適切な予防措置を実施し、適切な接地要件を考慮してください。

リアルタイム・モードでマイコンを停止させるには、次の手順に従います。

1. 400V DC入力をオフにし、数秒間待ちます。
2. ツールバーの“Halt”をクリックして、プロセッサを停止します。
3.  をクリックして、マイコンのリアルタイム・モードを終了します。
4. マイコンをリセットします。

9.2 ビルド2: 閉電圧ループ

目的

このビルドの目的は、CCS環境から完全なプッシュプル・プロジェクトの動作を検証することです。

概要

図 59 に、このビルドの各ソフトウェア・ブロックを示します。PWMおよびADCドライバ・ブロックは、この章のビルド1の場合と同様に使用されます。電圧または電流ループには、2極/2ゼロのコントローラを使用します。アプリケーションの制御ループ要件に応じて、PI、3極/3ゼロなど、他のコントローラ・ブロックも使用できます。

図 59に示されるとおり、電圧ループ・ブロックは100kHzで実行されます。CNTL2P2Zは、IIRフィルタ構造から実現される2次補償回路です。この機能は、すべてのペリフェラルから独立しており、CNF関数の呼び出しを必要としません。

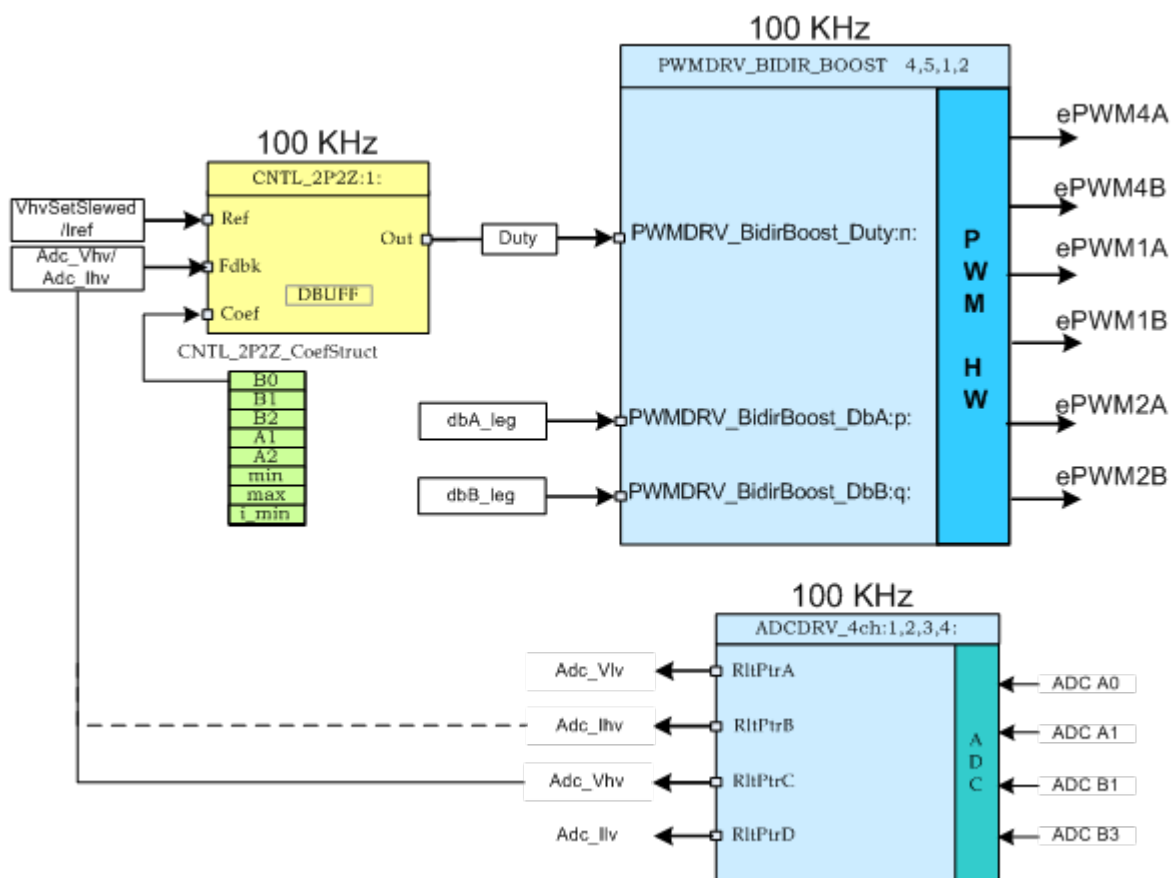


図 59. ビルド2のソフトウェア・ブロック

変更する5つの係数は、構造体CNTL_2P2Z_CoefStruct1の要素として格納されています。この構造体の他の要素によって、コントローラの出力がクランプされます。システムで複数のループが必要な場合、CNTL_2P2Zブロックは複数回インスタンス化することができます。各インスタンスには、それぞれ個別の係数のセットを使用できます。試行錯誤を通して5つの係数を直接独立に操作することはほぼ不可能であり、数学的な分析や、MATLAB、Mathcadなどの支援ツールが必要となります。これらのツールは、ボード・プロットや根軌跡など、位相マージンやゲイン・マージンを決定するための機能を提供します。

ループ調整をシンプルに保ち、複雑な計算や分析ツールを不要にするために、TIでは、B0、B1、B2、A1、A2に対して、より直観的な係数ゲインP、I、Dをマッピングすることで、係数選択問題の自由度を5から3へと減らしました。この自由度の減少により、P、I、Dがそれぞれ独立して段階的に調整されます。これらのマッピングの式を以下に示します。

補償回路ブロック (CNTL_2P2Z) は2つの極と2つのゼロを持ち、一般的なIIRフィルタ構造に基づいています。このブロックには、1つのリファレンス入力と1つの帰還入力があります。帰還は、VMCモードの選択時はセンスされた出力電圧 (Adc_Vhv) であり、ACMCモードの選択時はセンスされた出力電流 (Adc_Ihv) です。この選択は、Bi-dir-Settings.hファイルから実行できます。コントローラへのリファレンス入力は、モード選択に基づいて、出力電圧リファレンス・コマンド (Vref) のスルー・バージョン (VhvSetSlewed)、またはリファレンス出力電流コマンド (Iref) です。伝達関数は式 8 で与えられます。

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (8)$$

PIDコントローラの再帰形式は式 9 で与えられます。

$$u(k) = u(k-1) + b_0 e(k) + b_1 e(k-1) + b_2 e(k-2) \quad (9)$$

ここで、式 10、式 11、式 12 が成り立ちます。

$$b_0 = K_p' + K_i' + K_d' \quad (10)$$

$$b_1 = -K_p' + K_i' - 2 K_d' \quad (11)$$

$$b_2 = K_d' \quad (12)$$

また、Zドメイン伝達関数形式は式 13 のようになります。

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}} = \frac{b_0 z^{-2} + b_1 z + b_2}{z^2 - z} \quad (13)$$

これを一般的な形式と比較すると、PIDはCNTL_2P2Z制御の特殊なケースであり、式 14 が成り立つ場合です。

$$a_1 = -1 \text{ and } a_2 = 0 \quad (14)$$

P、I、およびD係数は、Pgain、Igain、およびDgainです。これらのP、I、およびD係数はQ26形式で使用されます。CCS監視ビューからの調整を単純化するために、これらの3つの係数をさらに0~999の値へと変換します (Pgain_Gui、Igain_Gui、Dgain_Gui)。また、ループ・パラメータも、外部の数学ツール (MATLAB、Mathcadなど) から得られる調整済みの値に基づいて、直接変更できます。ループ係数は、I5Q10形式の変数b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiを使用して直接変更でき、その後で、2P2Zコントローラ用に5個のQ26係数へと変換されます。

このプロジェクトでは、実行中に係数を簡単に切り替えて、両方のループ調整方法を簡単に評価できます。この切り替えは、GUI上で2P2Z(On)/PID(Off)ボタンをクリックするか、またはCCSから監視ビュー上でpid2p2z_GUI変数を0または1に変更することによって行えます。このプロジェクトでは、GUI環境からのPIDベースのループ調整 (pid2p2z_GUI = 0) を使用しています。pid2p2z_GUIを1に変更する前に、b2_Gui、b1_Gui、b0_Gui、a2_Gui、およびa1_Gui変数に信頼できる係数値がプログラミングされていることを確認してください。

注: このプロジェクトには、有効なb2_Gui、b1_Gui、b0_Gui、a2_Gui、およびa1_Guiパラメータ値が含まれていません。TIでは、pid2p2z_GUI = 0に保持することで、デフォルトのP、I、Dベースのループを使用することを推奨します。

9.2.1 手順

プロジェクトのビルドとロード

構成済みの作業環境を使用してこのビルドをすばやく実行するには、次の手順に従います。

1. 2 で示したとおりに、基板上のすべてのジャンパが正しく取り付けまたは取り外されていることを確認します。
2. F28035 controlCARDを100ピンDIMMコネクタに挿入します。
3. 12V DCのベンチ電源をTP10とTP11の間に正しい極性で接続します。
4. 400VのプログラマブルDC電源を400V入力コネクタに接続し、12V負荷を12Vコネクタに接続します (この負荷は基板の定格を超えないようにしてください)。

5. PCと基板の間にUSB-B/USB-Aケーブルを接続します。

注: この時点では、どの電源もオンにしないでください。

6. 基板をJTAG接続でテストするのが初めてである場合は、xds100v2-FT_Prog_v2.2.zipファイル内のprogram_ftdi.batファイルを実行して、基板上のFTDIチップをプログラミングします。
7. Code Composer Studio (CCSv5以降)を起動します。
8. CCSを画面上で最大化します。
9. Welcome画面が表示された場合は閉じます。

注: プロジェクトには、マイコンのハードウェアで実行できる実行形式出力ファイル(.out)を開発するために必要な、すべてのファイルとビルド・オプションが含まれています。

10. メニュー・バーで、“Project” → “Import Existing CCS/CCE Eclipse Project”の順にクリックします。
11. ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\v1_00_00_00\Boost_Modeディレクトリを選択します。
12. “Projects”タブで、Bi-dir_Boostにチェックマークが付いていることを確認します。
13. “Finish”をクリックします。

注: このプロジェクトは、すべての必要なツール(コンパイラ、アセンブラ、リンカ)を使用してビルドを実行します。

14. 左側の“Project”ウィンドウで、プロジェクトの左側にある“+”をクリックします。
15. メイン・ファイル内にあるビルド2の初期化コードを見つけます。
16. コードの内容を確認します。(このコードによって、すべての制御ブロックが設定、初期化され、制御フロー内で接続されます。)
17. Bi-dir-Settings.hファイルで、段階的ビルド・オプションとして2を選択します。

注: 以前に別のオプションをビルドした場合は、次の手順に従います。

1. プロジェクト名を右クリックします。
 2. “Clean Project”をクリックします。
 3. “Project” → “Build All”の順にクリックします。
 4. ビルド・ウィンドウでツールが実行されるのを確認します。
-

18. 12V DCベンチ電源をオンにします。
19. “Debug”ボタンをクリックします。

注: ビルド1のコードがコンパイルされてロードされます。

F28035ターゲット構成を作成するか、またはF28035.ccxmlファイルから構成をコピーするには、次の手順に従います。

1. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
2. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
3. “Run”→“Load”→“Load Program”の順に選択します。
4. Boost_Mode\F2803x_Flash\Bi-dir_Boost.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

5. main()の先頭でプログラムを停止します。

デバッグ環境ウィンドウ

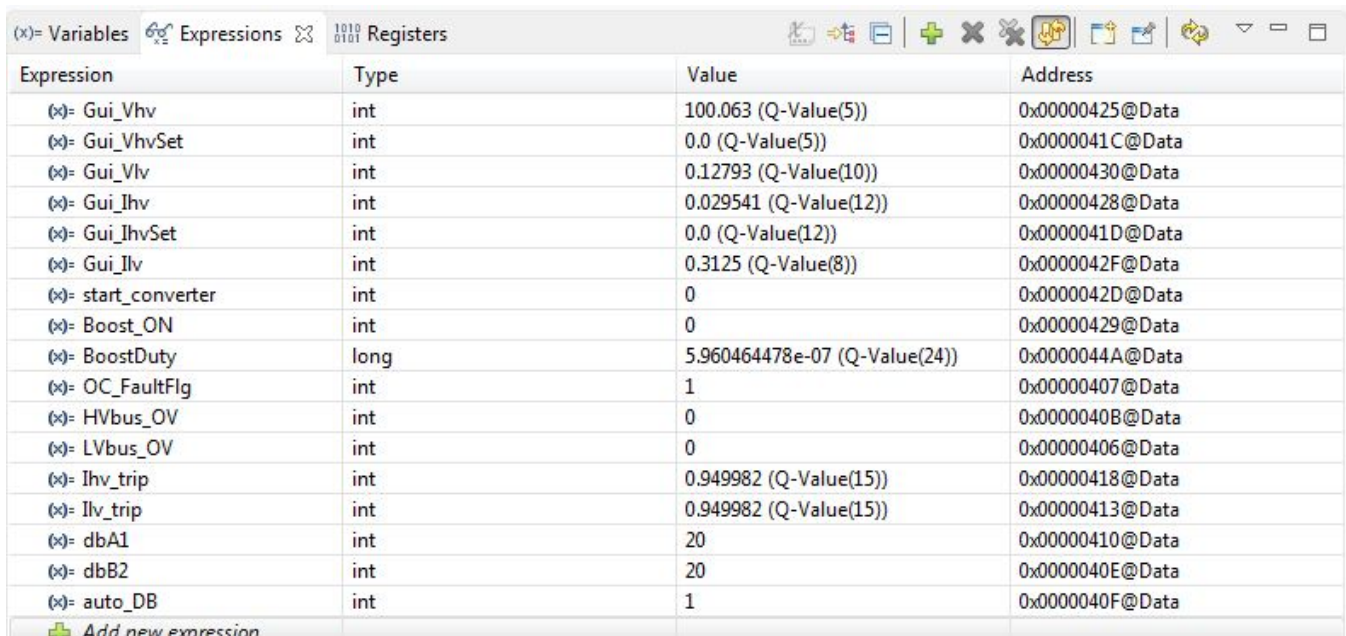
コードのデバッグ中に、ローカル変数およびグローバル変数を監視します。CCSのメモリ・ビューおよび監視ビューを使用できます。CCSでは、時間ドメインと周波数ドメインのプロットを作成できます。波形はグラフ・ウィンドウを使用して表示できます。

1. メニュー・バーで、“View” → “Scripting console”をクリックします。
2. スクリプト・コンソールの“Open File ()”コマンドを使用して、プロジェクト・ディレクトリから AddWatchWindowVars.js ファイルを開きます。

これにより、“Expressions”ウィンドウの各項目が設定されます。“Expressions”ウィンドウの外観については、[図 60](#)を参照してください。

いくつかの変数はメイン・コードのこの時点では初期化されておらず、不定の値が格納されています。

OC_FaultFlgがセットされている場合は、過電流状態によってPWM出力がシャットダウンされることを示しています。PWM出力は、デバイスがリセットされるまでこの状態に保持されます。Ipri_trip変数は、オンチップ・コンパレータ1に対する内部10ビットDACの基準レベルを設定します。これはQ15値です。[図 60](#)を参照してください。



Expression	Type	Value	Address
(x)= Gui_Vhv	int	100.063 (Q-Value(5))	0x00000425@Data
(x)= Gui_VhvSet	int	0.0 (Q-Value(5))	0x0000041C@Data
(x)= Gui_Vlv	int	0.12793 (Q-Value(10))	0x00000430@Data
(x)= Gui_Ihv	int	0.029541 (Q-Value(12))	0x00000428@Data
(x)= Gui_IhvSet	int	0.0 (Q-Value(12))	0x0000041D@Data
(x)= Gui_IlV	int	0.3125 (Q-Value(8))	0x0000042F@Data
(x)= start_converter	int	0	0x0000042D@Data
(x)= Boost_ON	int	0	0x00000429@Data
(x)= BoostDuty	long	5.960464478e-07 (Q-Value(24))	0x0000044A@Data
(x)= OC_FaultFlg	int	1	0x00000407@Data
(x)= HVbus_OV	int	0	0x0000040B@Data
(x)= LVbus_OV	int	0	0x00000406@Data
(x)= Ihv_trip	int	0.949982 (Q-Value(15))	0x00000418@Data
(x)= Ilv_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
(x)= dbA1	int	20	0x00000410@Data
(x)= dbB2	int	20	0x0000040E@Data
(x)= auto_DB	int	1	0x0000040F@Data
+ Add new expression			

図 60. 昇圧モードの“Expressions”ウィンドウ: ビルド2

コードの実行

1. リアルタイム・モードをイネーブルにします。
2. 監視ビューの連続的更新をイネーブルにし、監視ビューの連続的更新を変更します。
3. ツールバーの“Run”をクリックします。
4. start_converter変数を0.0 (Q24) に設定します。(この変数は電圧コマンドを規定します。)
5. DC出力のPSFBシステムに2kΩの抵抗性負荷を適用します。

注: TIでは、絶縁型DC電源を使用して、400VのDC入力を基板に供給することを推奨します。

6. 400V DC電源を100V DC出力に設定します。
7. この電源をオンにします。
8. 12V DC電源の出力を12Vに設定します。

9. この電源をオンにします。
10. `start_converter`を1に設定します。(400V DCバス電圧が約150Vまで上昇し、400V DC電源は0まで減少します。)
11. 監視ビューで`Gui_VhvSet`の値を200Vに設定して、電圧コマンドを増加させます。(昇圧動作が行われ、出力電圧は設定されたコマンドに従います。)
12. 出力電圧の上昇を観測します。

注: 出力電圧が基板の定格を超えないようにしてください。

`Gui_VhvSet`コマンドは、400V DCの電源電圧設定を下回らないようにしてください。

13. 異なる`Gui_VhvSet`値に対して、監視ビューで異なるADC結果を観測します。(この値が変更されるごとに、出力電圧はこの新しい値へと上昇または下降します。`VhvSlewRate`を変更することで、出力電圧の上昇または下降レートを変更します。)

図 61 は、入力電圧が12V、負荷が約2Ωのときの225Vでのシステム動作に対応する監視ビューを示しています。

Expression	Type	Value	Address
(x) Gui_Vhv	int	224.719 (Q-Value(5))	0x00000424@Data
(x) Gui_VhvSet	int	225.0 (Q-Value(5))	0x0000041C@Data
(x) Gui_Vlv	int	12.0098 (Q-Value(10))	0x0000042E@Data
(x) Gui_Ihv	int	0.35498 (Q-Value(12))	0x00000430@Data
(x) Gui_IhvSet	int	0.0 (Q-Value(12))	0x0000041D@Data
(x) Gui_Ilv	int	3.52344 (Q-Value(8))	0x0000042F@Data
(x) start_converter	int	1	0x00000428@Data
(x) Boost_ON	int	1	0x0000042A@Data
(x) BoostDuty	long	0.30097121 (Q-Value(24))	0x00000450@Data
(x) OC_FaultFlg	int	0	0x00000407@Data
(x) HVbus_OV	int	0	0x0000040B@Data
(x) LVbus_OV	int	0	0x00000406@Data
(x) Ihv_trip	int	0.949982 (Q-Value(15))	0x00000418@Data
(x) Ilv_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
(x) dbA1	int	32	0x00000411@Data
(x) dbB2	int	18	0x0000040E@Data
(x) auto_DB	int	1	0x0000040C@Data
+ Add new expression			

図 61. 昇圧モードの“Expressions”ウィンドウ: ビルド2のVMCモード(ランタイム)

図 62 は、入力電圧が12V、負荷が約2kΩのときの325V出力でのシステム動作に対応する監視ビューを示しています。(12Vコネクタには100Ωの負荷も接続されています。)

Expression	Type	Value	Address
(x)- Gui_Vhv	int	325.438 (Q-Value(5))	0x00000424@Data
(x)- Gui_VhvSet	int	325.0 (Q-Value(5))	0x0000041C@Data
(x)- Gui_Vlv	int	11.8809 (Q-Value(10))	0x0000042E@Data
(x)- Gui_Ihv	int	0.470947 (Q-Value(12))	0x00000430@Data
(x)- Gui_IhvSet	int	0.0 (Q-Value(12))	0x0000041D@Data
(x)- Gui_IlV	int	6.48047 (Q-Value(8))	0x0000042F@Data
(x)- start_converter	int	1	0x00000428@Data
(x)- Boost_ON	int	1	0x0000042A@Data
(x)- BoostDuty	long	0.5029316545 (Q-Value(24))	0x00000450@Data
(x)- OC_FaultFlg	int	0	0x00000407@Data
(x)- HVbus_OV	int	0	0x0000040B@Data
(x)- LVbus_OV	int	0	0x00000406@Data
(x)- Ihv_trip	int	0.949982 (Q-Value(15))	0x00000418@Data
(x)- Ilv_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
(x)- dbA1	int	32	0x00000411@Data
(x)- dbB2	int	18	0x0000040E@Data
(x)- auto_DB	int	1	0x0000040C@Data
+ Add new expression			

図 62. 昇圧モードの“Expressions”ウィンドウ: ビルド2、Vout = 325V

14. 負荷を変化させたときの出力電圧および入力電流への影響を観測します。(出力電圧には影響がないはずですが。)

注: 電圧が、このドキュメントの仕様セクションに示される基板の能力を超えないようにしてください。


15. 入力電圧(12V DC電源)の変化による影響を観測します。

注: 位相は小さい間隔で増加させます。常に出力電圧を慎重に観測してください。電圧が基板の定格を超えないようにしてください。

オシロスコープを使用して、PWMゲート駆動信号、入力電圧、電流、出力電圧などの波形をプローブできます。

この絶縁型DC-DCコンバータに対してこれらの高電圧および高電流をプローブする際には、安全性に関する適切な予防措置を実施し、適切な接地要件を考慮してください。

リアルタイム・モードでマイコンを停止させるには、次の手順に従います。

1. 400V DC入力をオフにします。
2. 数秒間待ちます。
3. ツールバーの“Halt”をクリックして、プロセッサを停止します。
4.  をクリックして、マイコンのリアルタイム・モードを終了します。
5. マイコンをリセットします。

コンバータを出力インダクタ制御モード (ACMCモード) で制御するには、次の手順に従います。

1. Bi-dir-Settings.hファイルでVMC0_ACMC1を1に設定します。

注: 以前に別のオプションをビルドした場合は、次の手順に従います。

1. プロジェクト名を右クリックします。
 2. “Clean Project”をクリックします。
 3. “Project” → “Build All”の順にクリックします。
 4. ビルド・ウィンドウでツールが実行されるのを確認します。
-

2. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
3. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
4. “Run” → “Load” → “Load Program”の順に選択します。
5. Boost_Mode\F2803x_Flash\Bi-dir_Boost.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

6. main()の先頭でプログラムを停止します。
7. リアルタイム・モードをイネーブルにします。
8. 監視ビューの連続的更新をイネーブルにし、連続的更新間隔を変更します。
9. ツールバーの“Run”をクリックして、コードを実行します。
10. start_converterを0に設定します。
11. Gui_VhvSetを0.0(Q5)に設定します。(この変数は電圧コマンドを規定します。)
12. HV(400V) DC出力に2kΩの抵抗性負荷を適用します。
13. 400V入力を100V DC出力に設定します。
14. この電源をオンにします。
15. 12V DC電源を12Vに設定します。
16. この電源をオンにします。

17. start_converterを1に設定します。(400V DCバス電圧が約150Vまで上昇し、400V DC電源電流は0まで減少します。)
18. 監視ウィンドウに表示されるGui_Ihvhの値を観測し、書き留めます(Q12)。
19. Gui_IhvhSetコマンドの値を0から、手順18で書き込んだGui_Ihvh値の0.01 (Q12) 以下の値へと変更します。

CAUTION

Gui_IhvhSetコマンドを増加させる幅は、1回に0.01 (Q12) 未満としてください。

コマンドの小さな変更でも、負荷によっては400V出力で非常に高い電圧変化につながる場合があります。


ACMCモードでテストする際には注意が必要です。

図 63 は、入力電圧が約11.8V、負荷が2k Ω 、結果の出力が270Vのときの、0.4Aの電流コマンドに対するシステム動作の監視ビューを示しています。(12Vコネクタには100 Ω の負荷も接続されています。)

Expression	Type	Value	Address
(x)- Gui_Vhvh	int	269.281 (Q-Value(5))	0x00000423@Data
(x)- Gui_VhvhSet	int	0.0 (Q-Value(5))	0x0000041B@Data
(x)- Gui_Vlv	int	11.7627 (Q-Value(10))	0x0000042D@Data
(x)- Gui_Ihvh	int	0.400879 (Q-Value(12))	0x0000042F@Data
(x)- Gui_IhvhSet	int	0.399902 (Q-Value(12))	0x0000041C@Data
(x)- Gui_Ilv	int	4.80078 (Q-Value(8))	0x0000042E@Data
(x)- start_converter	int	1	0x00000408@Data
(x)- Boost_ON	int	1	0x00000429@Data
(x)- BoostDuty	long	0.4147862196 (Q-Value(24))	0x0000044E@Data
(x)- OC_FaultFlg	int	0	0x00000407@Data
(x)- HVbus_OV	int	0	0x0000040B@Data
(x)- LVbus_OV	int	0	0x00000406@Data
(x)- Ihvh_trip	int	0.949982 (Q-Value(15))	0x00000427@Data
(x)- Ilvh_trip	int	0.949982 (Q-Value(15))	0x00000413@Data
(x)- dbA1	int	32	0x00000411@Data
(x)- dbB2	int	18	0x0000040E@Data
(x)- auto_DB	int	1	0x0000040C@Data
+ Add new expression			

図 63. 昇圧モードの“Expressions”ウィンドウ: ビルド2のACMCモード(ランタイム)

リアルタイム・モードでマイコンを停止させるには、次の手順に従います。

1. 400V電源(100Vに設定)をオフにし、数秒間待ちます。
2. 12V入力をオフにし、数秒間待ちます。
3. ツールバーの“Halt”をクリックして、プロセッサを停止します。
4.  をクリックして、マイコンのリアルタイム・モードを終了します。
5. マイコンをリセットします。
6. CCSを閉じます。

10 双方向DC-DC(フル) – ソフトウェアの概要

10.1 ソフトウェア制御フロー

この実装では、両方向への電力の流れを制御でき、ソフトウェア・コマンドに基づいて降圧モードと昇圧モードをその場で切り替えることができます。降圧および昇圧変換にはVMCモードが使用されます。このプロジェクトは、降圧モード・プロジェクトのビルド・レベル2(6.2を参照)と昇圧モード・プロジェクトのビルド・レベル2(9.2を参照)を組み合わせたもので、さらに2つのモード間の高速でスムーズな切り替えを実現するために適切なコードを追加しています。

Bi-dir_Fullプロジェクトでは、Cバックグラウンド/ASM-ISRフレームワークを使用します。このフレームワークは、アプリケーションのメインのサポート・プログラムとしてCコードを使用し、すべてのシステム管理タスク、意思決定、インテリジェンス、およびホストとのやり取りを実行します。アセンブリ・コードは、すべての重要な制御コードを実行するISRのみに厳密に制限されています。これには一般に、ADCの読み取り、制御用の計算、およびPWMとDACの更新が含まれます。図64に、このプロジェクトの全般的なソフトウェア・フローを示します。

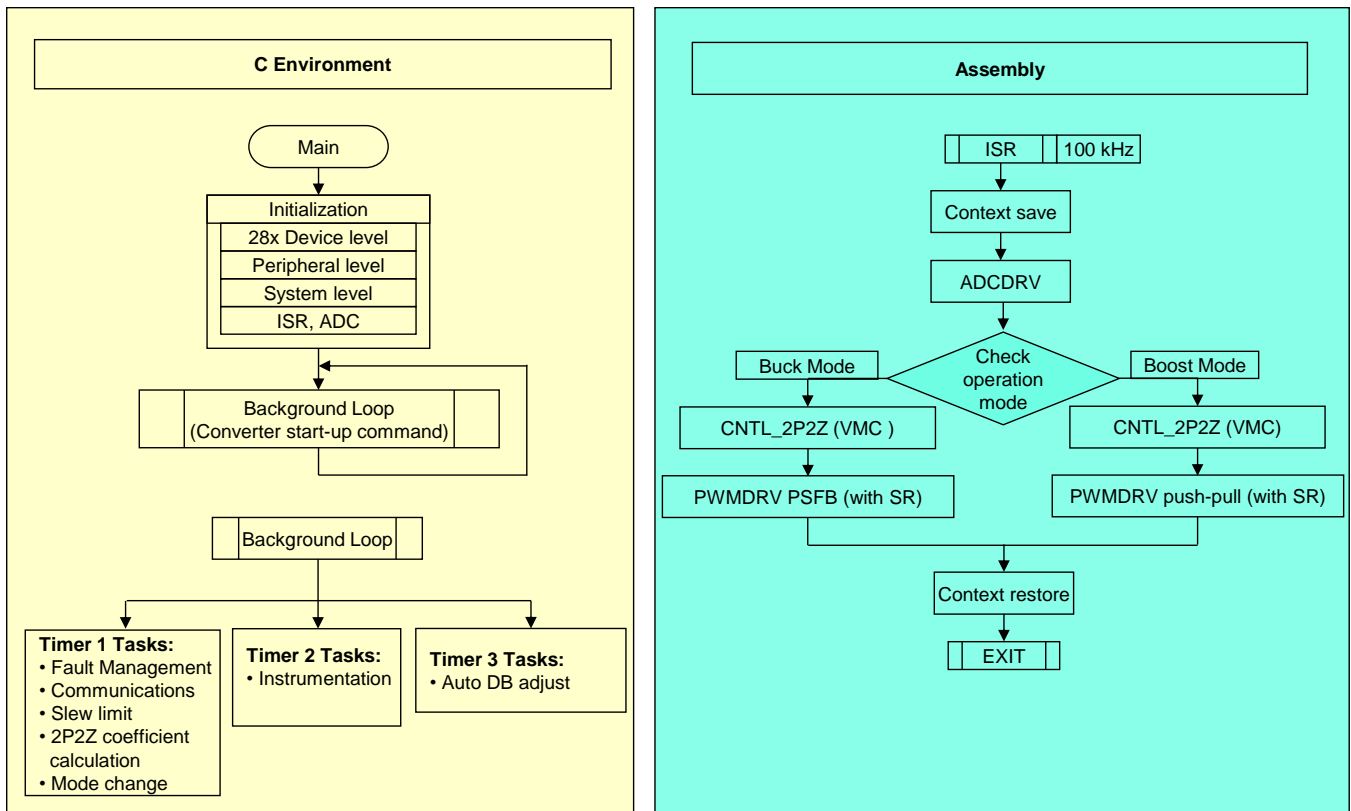


図 64. 昇圧モードのソフトウェア・フロー

このプロジェクトの主要なフレームワークCファイルを以下に示します。

- Bi-dir-Main.c – このファイルはアプリケーションを初期化、実行、および管理します。
- Bi-dir-DevInit_F2803x.c – このファイルは、マイコンを1回のみ初期化および構成し、クロック、PLL、GPIOの設定などの機能を含みます。

ISRは、次の1つのファイルから構成されます。

- Bi-dir-DPL-ISR.asm – このファイルは、時間が重要となる制御タイプ・コードをすべて含んでいます。このファイルには初期化セクション(1回実行)とランタイム・セクション(PWMスイッチング周波数で実行)が含まれています。

Power Libraryの関数(モジュール)は、このフレームワークから呼び出されます。

ライブラリ・モジュールにはCコンポーネントとアセンブリ・コンポーネントの両方が含まれる場合があります。表 7に、Cモジュール名および対応するアセンブリ・モジュール名を示します。

表 7. ライブラリ・モジュール

C構成関数	ASM初期化マクロ	ASMランタイム・マクロ
DAC_Cnf.c		
ADC_SOC_Cnf.c	ADCDRV_4CH_INIT m,n,p,q	ADCDRV_4CH m,n,p,q
PWM_PSFV_VMC_SR_Cnf.c	PWMDRV_PSFV_VMC_SR_INIT m,n,p	PWMDRV_PSFV_VMC_SR m,n,p
PWM_PSFV_VMC_SR_Cnf.c	PWMDRV_BIDIR_BOOST_INIT m,n,p,q	PWMDRV_BIDIR_BOOST m,n,p,q
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

図 65 に制御ブロックを示します。

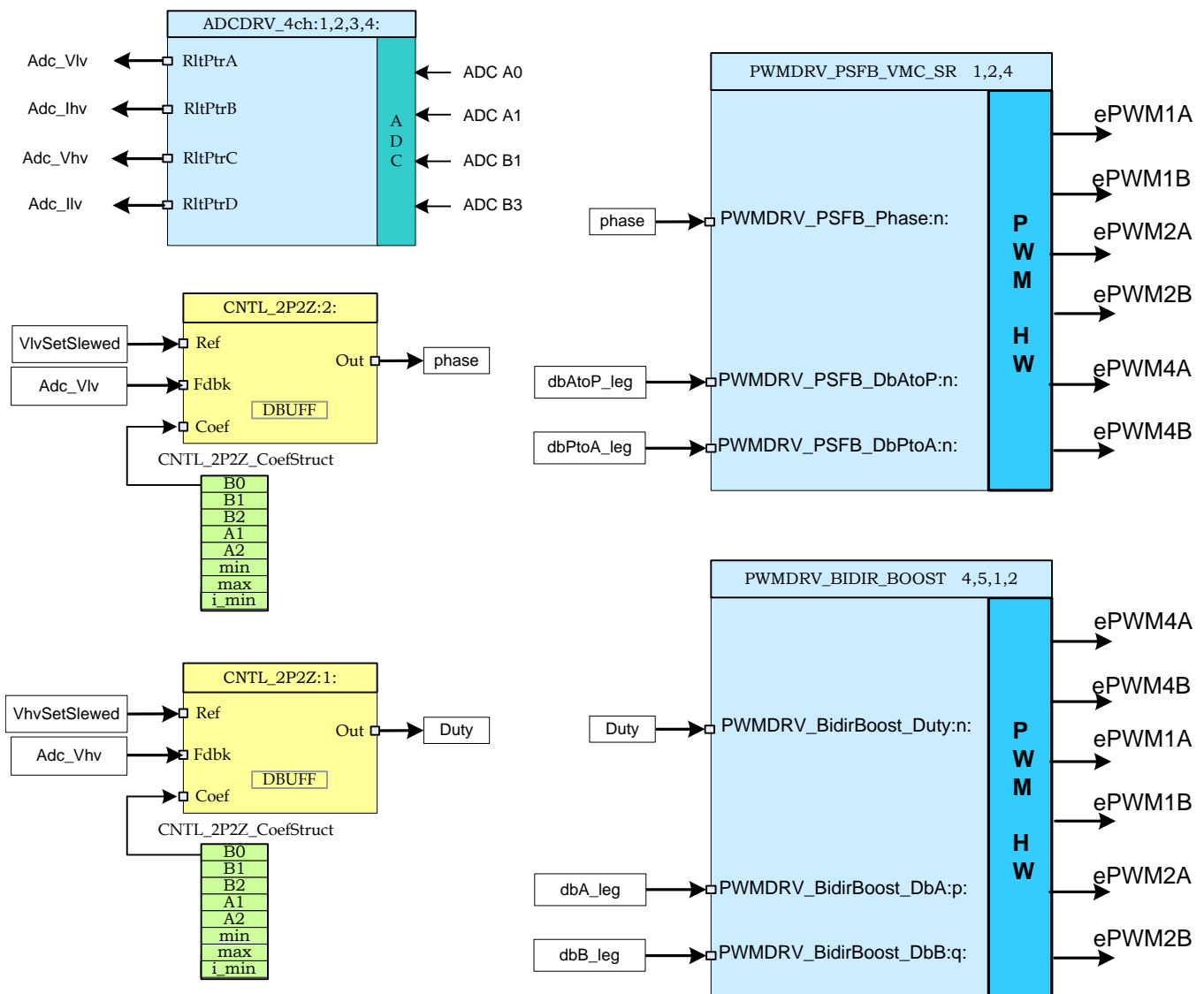


図 65. ソフトウェア・ブロック

図 66では、濃い青色のブロックがC2000マイコン上のハードウェア・モジュールを表しています。青色のブロックは、これらのモジュール用のソフトウェア・ドライバです。黄色のブロックは、制御ループ用のコントローラ・ブロックです。ここでは2極/2ゼロのコントローラを使用していますが、PI/PIDや3極/3ゼロなど、このアプリケーションに適切に実装可能な他の種類のコントローラも使用できます。図 66に示すように、このようなモジュール型のライブラリ構造によって、システム全体のソフトウェアの流れが見やすく、理解しやすくなっています。また、この構造により、各種の機能の使用、追加、削除が簡単になります。

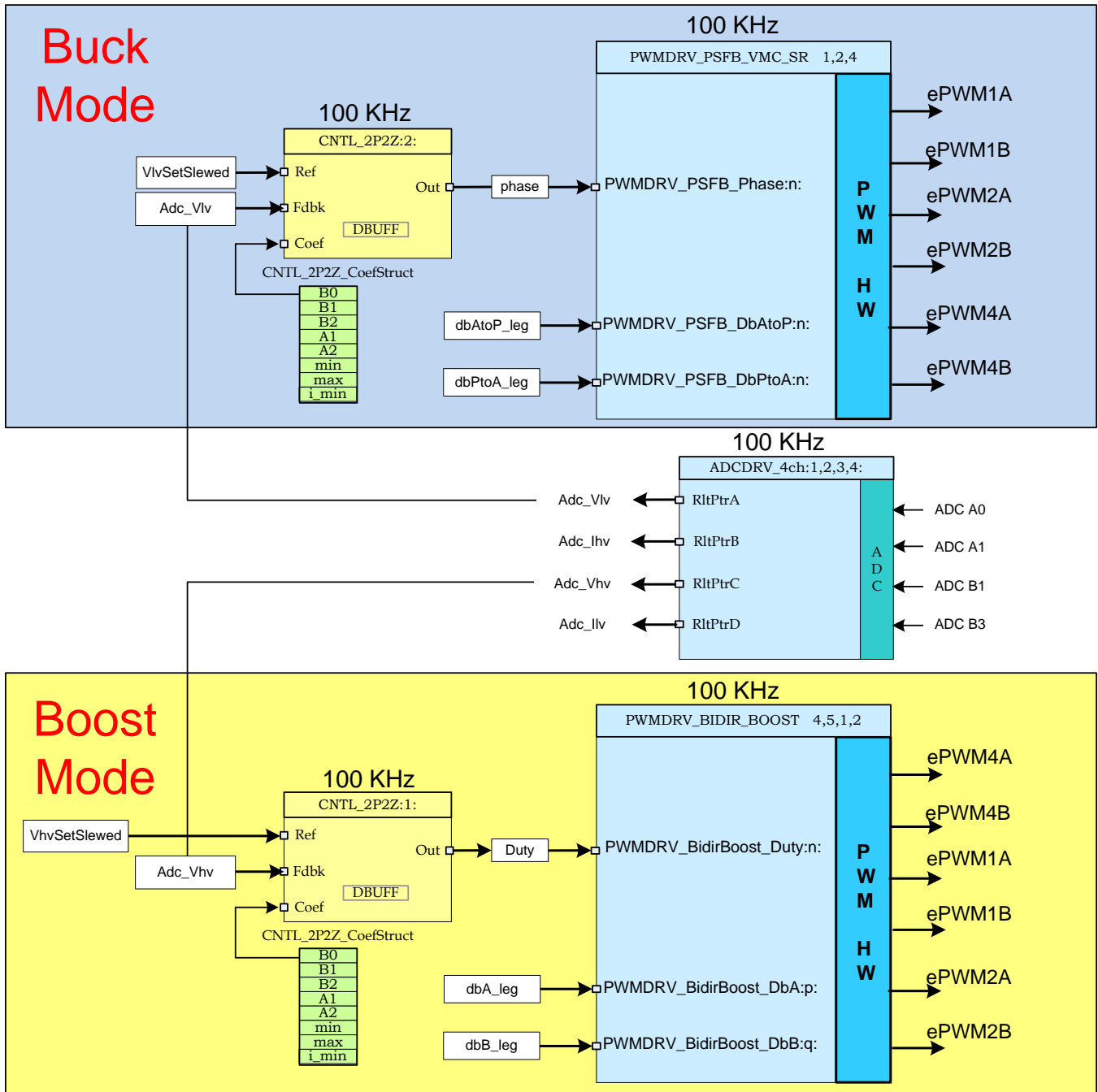


図 66. 制御フロー

システムは、降圧モードと昇圧モードの両方で1つの電圧帰還ループによって制御されます。システムが一方のモードのときは、そのモードでの動作に必要なソフトウェア・ブロックだけが実行され、他方のモードにのみ必要なブロックは実行されません。図 66には、制御ブロックの実行レートも示されています。たとえば、電圧コントローラは100kHzのレート(PWMスイッチング周波数と同じ)で実行されます。図 66で実装されている制御について次に説明します。

センスされた出力電圧(Adc_Vhv/Adc_Vlv)は、電圧コントローラの電圧リファレンスコマンドのスルー・バージョン(VhvSetSlewed/VlvSetSlewed)と比較されます。コントローラの出力によって、昇圧モードでは2つのプッシュプル・スイッチを駆動するPWM信号のデューティ・サイクルを直接制御し、降圧モードではフルブリッジ・スイッチを駆動するPWM信号間の位相シフトを直接制御します。これらのPWMドライバ・モジュールは、両方のモードで同期整流器スイッチも駆動します。

モード遷移

2つのモード間で高速でシームレスな遷移を実現することが重要です。これらの遷移は、ステートマシン・タスクA3内のモード遷移コードによって実現されます。(変数Buck0_Boost1を1に設定して)降圧から昇圧へのモード遷移が開始されると、モード遷移コードによって降圧モード・シャットダウン・ルーチンが実行され、出力電圧コマンド(Gui_VhvSet)をHVバス上の前の電圧(Gui_Vhv)よりも約4V高い値に設定した状態で、コンバータが昇圧モードで起動します。同様に、(変数Buck0_Boost1を0に設定して)昇圧から降圧へのモード遷移が開始されると、モード遷移コードによって昇圧モード・シャットダウン・ルーチンが実行され、出力電圧コマンド(Gui_VlvSet)をLVバス上の前の電圧(Gui_Vlv)よりも約0.5V高い値に設定した状態で、コンバータが降圧モードで起動します。

保護

このプロジェクトのシャットダウン・メカニズムでは、オンチップのアナログ・コンパレータ1を使用して、トランスの高電圧巻線電流に対する過電流保護を実装しています。LVインダクタ電流の過電流保護は、オンチップ・アナログ・コンパレータ2を使用して実装されます。基準トリップ・レベルは内部の10ビットDACを使用して設定され、これらのコンパレータの反転端子に供給されます。コンパレータの出力は、センスされた電流が設定制限値を上回るときにePWM1、ePWM2、およびePWM4にワンショットのトリップ・アクションを生成するよう構成されています。LV側とHV側両方の過電圧保護および低電圧保護は、低速ステートマシン・タスクA1内部のソフトウェアに実装されています。過電圧または低電圧状態が検出されると、ePWM1、ePWM2、およびePWM4でワンショット・トリップ・アクションが開始されます。C2000デバイスのトリップ・メカニズムの柔軟性により、異なるトリップ・イベントに対して異なるアクションを実行することが可能になっています。このプロジェクトでは、電源段を保護するために、ePWM1A、ePWM1B、ePWM2A、ePWM2B、ePWM4A、およびePWM4Bが直ちにLowになります。これらの出力は、デバイスのリセットが実行されるまでこの状態に保持されます。

11 双方向DC-DC(フル) – 手順

メイン・ソース・ファイル、ISRアセンブリ・ファイル、およびシステムを起動するCフレームワーク用のプロジェクト・ファイルは、次のディレクトリに格納されています(最新バージョンのソフトウェア・パッケージを使用してください)。

`..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12lv1_00_00_00\Bi_Directional_Full`

図 7 に、これらのテスト用のハードウェア構成を示します。

WARNING

基板上には高電圧が印加されています。ラボ環境で経験を積んだ電源技術者のみが基板を取り扱う必要があります。この基板を安全に評価するには、適切な絶縁型の高電圧DC電源を使用してください。基板にDC電源を印加する前に、電圧計および適切な抵抗性負荷または電子負荷を出力に接続する必要があります。電源が印加されている間は、基板に触れないでください。

Bi_Directional_Fullソフトウェア内のサンプルをビルドして実行する方法の手順を以下に示します。

目的

このプロジェクトの目的は、このシステムの双方向動作、および降圧および昇圧動作モード間でのすばやい遷移について評価することです。このセクションでは、プロジェクトのビルドと実行の手順について説明します。

概要

図 66 に、このプロジェクトの各ソフトウェア・ブロックを示します。降圧および昇圧モード制御ブロックは、6.2および9.2と同様に使用されます。電圧ループには、2極/2ゼロのコントローラを使用します。アプリケーションの制御ループ要件に応じて、PI、3極/3ゼロなど、他のコントローラ・ブロックも使用できます。図 66 に示されるとおり、すべてのソフトウェア・ブロックは100kHzで実行されます。

ループ調整をシンプルに保ち、複雑な計算や分析ツールを不要にするために、B0、B1、B2、A1、A2に対して、より直観的な係数ゲインP、I、Dをマッピングすることで、係数選択問題の自由度を5から3へと減らしました。

これらのP、I、D係数は、昇圧モードではPgainhv、Igainhv、Dgainhv、降圧モードではPgainlv、Igainlv、Dgainlvです。これらのP、I、およびD係数はQ26形式で使用されます。CCS監視ビューからの調整を単純化するために、これらの3つの係数をさらに0~999の値へと変換します(Pgainhv_Gui、Igainhv_Gui、Dgainhv_Gui、Pgainlv_Gui、Igainlv_Gui、Dgainlv_Gui)。また、ループ・パラメータも、外部の数学ツール(MATLAB、Mathcadなど)から得られる調整済みの値に基づいて、直接変更できます。ループ係数は変数b2hv_Gui、b1hv_Gui、b0hv_Gui、a2hv_Gui、a1hv_Gui、b2lv_Gui、b1lv_Gui、b0lv_Gui、a2lv_Gui、a1lv_GuiをI5Q10形式で使用することにより、直接変更できます。これらはさらに、各2P2Zコントローラに対する5個のQ26係数に変換されます。

このプロジェクトでは、実行中に係数を簡単に切り替えて、両方のループ調整方法を簡単に評価できます。実行中に係数を切り替えるには、CCSから監視ビューでpid2p2zhv_GUIおよびpid2p2zlv_GUI変数を0または1に変更します。このプロジェクトでは、GUI環境からのPIDベースのループ調整を使用しています。pid2p2zhv_GUIまたはpid2p2zlv_GUI変数を1に変更する前に、b2hv_Gui、b1hv_Gui、b0hv_Gui、a2hv_Gui、a1hv_Gui、b2lv_Gui、b1lv_Gui、b0lv_Gui、a2lv_Gui、a1lv_Gui変数に信頼できる係数値がプログラミングされていることを確認してください。

注: このプロジェクトには、有効なb2hv_Gui、b1hv_Gui、b0hv_Gui、a2hv_Gui、a1hv_Gui、b2lv_Gui、b1lv_Gui、b0lv_Gui、a2lv_Gui、a1lv_Guiパラメータ値が含まれていません。TIでは、pid2p2zhv_GUI = 0およびpid2p2zlv_GUI = 0に保持することで、デフォルトのP、I、Dベースのループを使用することを推奨します。

11.1 手順

CCSを起動し、プロジェクトを開く

このビルドをすばやく実行するには、次の手順に従います。

1. 2で示したとおりに、基板上のすべてのジャンパが正しく取り付けまたは取り外されていることを確認します。
2. F28035 controlCARDを100ピンDIMMコネクタに挿入します。
3. 12V DCのベンチ電源をTP10とTP11の間に正しい極性で接続します。
4. 400VのプログラマブルDC電源を400V入力コネクタに接続し、12V負荷を12Vコネクタに接続します(この負荷は基板の定格を超えないようにしてください)。
5. PCと基板の間にUSB-B/USB-Aケーブルを接続します。

注: この時点では、どの電源もオンにしないでください。

6. 基板をJTAG接続でテストするのが初めてである場合は、xds100v2-FT_Prog_v2.2.zipファイル内のprogram_ftdi.batファイルを実行して、基板上のFTDIチップをプログラミングします。
7. Code Composer Studio (CCSv5以降)を起動します。
8. CCSを画面上で最大化します。
9. Welcome画面が表示された場合は閉じます。

注: プロジェクトには、マイコンのハードウェアで実行できる実行形式出力ファイル(.out)を開発するために必要な、すべてのファイルとビルド・オプションが含まれています。

10. メニュー・バーで、“Project” → “Import Existing CCS/CCE Eclipse Project”の順にクリックします。
11. ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12lv1_00_00_00\Bi_Directional_Fullディレクトリを選択します。
12. “Projects”タブで、Bi-dir_Fullにチェックマークが付いていることを確認します。
13. “Finish”をクリックします。

注: このプロジェクトは、すべての必要なツール(コンパイラ、アセンブラ、リンカ)を使用してビルドを実行します。

14. 左側の“Project”ウィンドウで、プロジェクトの左側にある“+”をクリックします。図 67を参照してください。

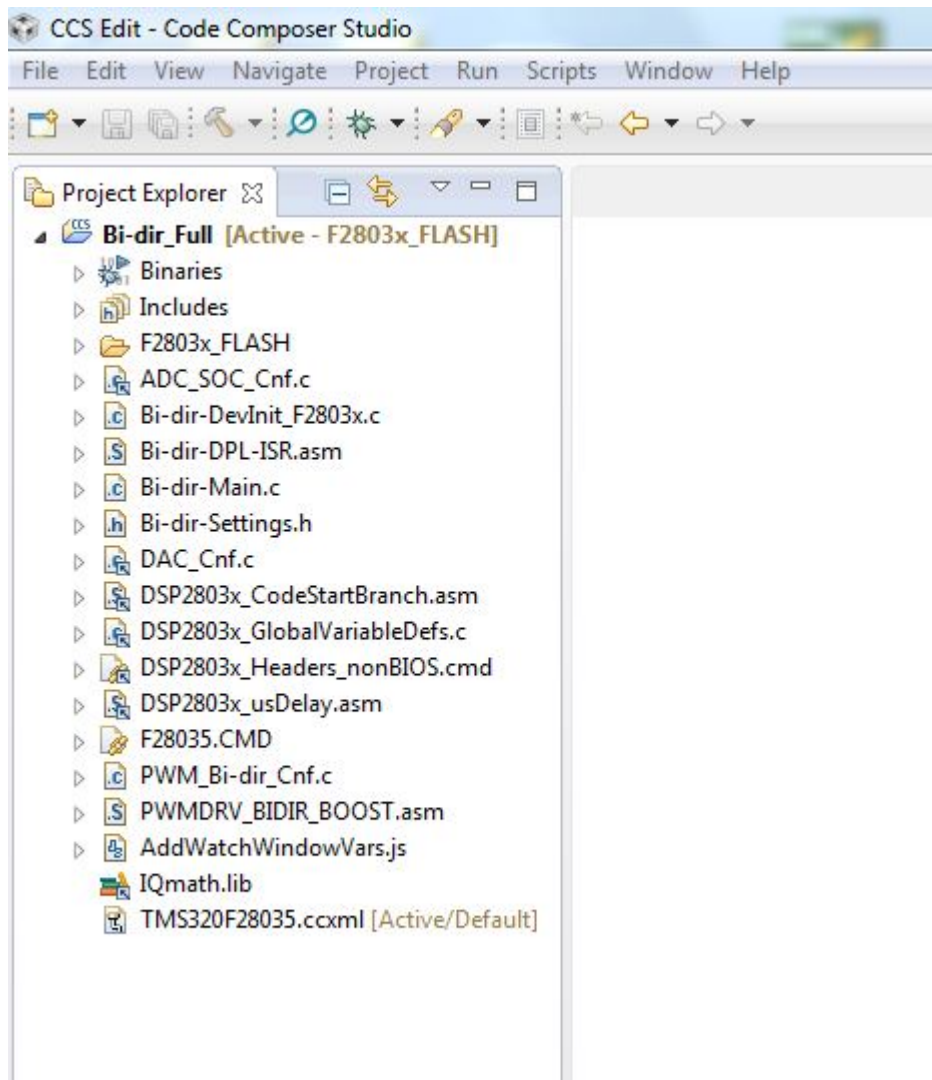


図 67. 完全な双方向プロジェクトのCCSプロジェクト・ウィンドウ

15. メイン・ファイル内にあるビルド2の初期化コードを見つけてます。
 16. コードの内容を確認します。(このコードによって、すべての制御ブロックが設定、初期化され、制御フロー内で接続されます。)

デバイスの初期化、メイン、およびISRファイル

注: ソース・ファイルには一切変更を加えないでください。

1. Bi-dir-DevInit_F2803x.cをダブルクリックします。
2. システム・クロック、ペリフェラル・クロックのプリスケール、およびペリフェラル・クロックのイネーブルが設定されていることを確認します。
3. 共有GPIOピンが設定されていることを確認します。
4. Bi-dir-Main.cをダブルクリックします。
5. DeviceInit()関数の呼び出し、各段階的ビルド・オプションのコード、ISR初期化、およびバックグラウンドfor(;;)ループを表示します。
6. メイン・ファイル内で、PWMDRV_PSFV_VMC_SR、PWMDRV_BIDIR_BOOST、CNTL_2P2Z1(昇圧)、CNTL_2P2Z2(降圧)、およびADCDRV_4CHソフトウェア・ライブラリ・ブロックを制御フロー内で接続および初

期化するコードを見つけます。

7. Bi-dir-DPL-ISR.asmファイルを開きます。
8. _DPL_Initおよび_DPL_ISRセクションに対するBi-dir-DPL-ISR.asmを調べます。

注: PWMおよびADCドライバ・マクロのインスタンス化は、それぞれ初期化およびランタイム用に_DPL_Initおよび_DPL_ISRセクションで実行されます。確認を終えたファイルは閉じてかまいません。

9. ファイルを閉じます。

プロジェクトのビルドとロード

1. Bi-dir-Settings.hファイルで、段階的ビルド・オプションとして1を選択します。

注: 以前に別のオプションをビルドした場合は、プロジェクト名を右クリックして、“Clean Project”をクリックします。

2. “Project” → “Build All”の順にクリックします。
3. 12V DCベンチ電源をオンにします。
4. “Debug”ボタンをクリックします。

注: ビルド1のコードがコンパイルされてロードされます。

F28035ターゲット構成を作成するか、またはF28035.ccxmlファイルから構成をコピーするには、次の手順に従います。

1. “Target Configurations”ウィンドウで、F28035.ccxmlを右クリックします。
2. “Debug”ウィンドウでデバイスの名前を右クリックして、デバイスを接続します。
3. “Run”→“Load”→“Load Program”の順に選択します。
4. ..\Bi_Directional_Full\F2803x_FLASH\Bi-dir_Full.outを選択して、コードをロードします。

注: 右上の“CCS Debug”アイコンに、プログラムが“Debug Perspective”ビューであると表示されている必要があります。

5. main()の先頭でプログラムを停止します。

デバッグ環境ウィンドウ

コードのデバッグ中に、ローカル変数およびグローバル変数を監視します。CCSのメモリ・ビューおよび監視ビューを使用できます。CCSでは、時間ドメインと周波数ドメインのプロットを作成できます。波形はグラフ・ウィンドウを使用して表示できます。

1. メニュー・バーで、“View” → “Scripting console”をクリックします。
2. スクリプト・コンソールの“Open File ()”コマンドを使用して、プロジェクト・ディレクトリからAddWatchWindowVars.jsファイルを開きます。

これにより、“Expressions”ウィンドウの各項目が設定されます。“Expressions”ウィンドウの外観については、[図 68](#)を参照してください。

注: いくつかの変数はメイン・コードのこの時点では初期化されておらず、不定の値が格納されています。

OC_FaultFlgがセットされている場合は、過電流状態によってPWM出力がシャットダウンされることを示しています。同様に、HVbus_OV、LVbus_OV、HVbus_UV、LVbus_UVは、PWM出力をシャットダウンする過電圧および低電圧状態を示します。PWM出力は、デバイスがリセットされるまでこの状態に保持されます。Ihv_trip変数はオンチップ・コンパレータ1に対する内部10ビットDAC基準レベル (HV巻線の過電流スレッショルド)を設定し、Ilv_trip変数はオンチップ・コンパレータ2に対する内部10ビットDAC基準レベル (LVインダクタの過電流スレッショルド)を設定します。これらはQ15値です。図 68を参照してください。

Expression	Type	Value	Address
(x)- Gui_Vhv	int	102.219 (Q-Value(5))	0x0000043E@Data
(x)- Gui_VhvSet	int	0.0 (Q-Value(5))	0x00000421@Data
(x)- Gui_Vlv	int	0.078125 (Q-Value(10))	0x0000043A@Data
(x)- Gui_VlvSet	int	9.0 (Q-Value(10))	0x00000420@Data
(x)- Gui_Ihv	int	0.0297852 (Q-Value(12))	0x0000043B@Data
(x)- Gui_Ilsv	int	0.304688 (Q-Value(8))	0x00000427@Data
(x)- Buck0_Boost1	int	0	0x00000431@Data
(x)- start_converter	int	0	0x00000436@Data
(x)- Boost_ON	int	1	0x0000043D@Data
(x)- Buck_ON	int	0	0x00000435@Data
(x)- phase	long	0.0 (Q-Value(24))	0x00000472@Data
(x)- BoostDuty	long	5.960464478e-07 (Q-Value(24))	0x0000046C@Data
(x)- OC_FaultFlg	int	0	0x0000040B@Data
(x)- HVbus_OV	int	0	0x00000432@Data
(x)- LVbus_OV	int	0	0x0000040A@Data
(x)- HVbus_UV	int	0	0x00000430@Data
(x)- LVbus_UV	int	0	0x00000433@Data
(x)- Ihv_trip	int	0.949982 (Q-Value(15))	0x0000041D@Data
(x)- Ilv_trip	int	0.949982 (Q-Value(15))	0x0000041C@Data
(x)- dbA1	int	20	0x00000410@Data
(x)- dbB2	int	20	0x00000413@Data
(x)- dbAtoP_leg	int	20	0x00000414@Data
(x)- dbPtoA_leg	int	20	0x00000415@Data
(x)- auto_DB	int	1	0x00000411@Data
(x)- SR_mode	int	2	0x00000417@Data
+ Add new expression			

図 68. 完全なプロジェクトの“Expressions”ウィンドウ

リアルタイム・エミュレーションの使用

リアルタイム・エミュレーションは、マイコンの動作中にCCS内のウィンドウを最大10Hzのレートで更新できるようにする特別なエミュレーション機能です。このエミュレーションにより、グラフや監視ビューが更新され、また、マイコンの動作に影響を与える監視またはメモリ・ウィンドウ内の値を変更することができます。このエミュレーションは、制御パラメータをその場で調整する際に便利です。

リアルタイム・モードをイネーブルにするには、次の手順に従います。



1. 水平ツールバー上のボタンにマウス・ポインタを置きます。
2.  **Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)** をクリックします。

注: メッセージ・ボックスが表示されたら、“YES”を選択して、デバッグ・イベントをイネーブルにします。これにより、ステータス・レジスタ1 (ST1) のビット1 (DGBMビット) が0に設定されます。

DGBMは、デバッグ・イネーブル・マスク・ビットです。DGBMビットが0に設定されているときは、メモリおよびレジスタの値をホスト・プロセッサに渡して、デバッガ・ウィンドウを更新できます。

開いているまたは更新中のウィンドウや変数の数が多すぎる場合には、連続的に更新を実行すると、エミュレーション・リンクの帯域幅が限られているために、更新頻度が遅くなる場合があります。

“Expressions”ウィンドウの更新レートを下げるには、次の手順に従います。

1. “Expressions”ウィンドウで  を右クリックします。
2. “Continuous Refresh Interval...”を選択します。
3. 連続更新間隔の値を変更します。(通常、これらのテストには、1000msのレートで十分です。)
4. 監視ビューの  をクリックします。

コードの実行

1. ツールバーの“Run”をクリックします。
2. 監視ビューでstart_converter変数を0に設定します。
3. Buck0_Boost1を0に設定します。
4. HV (400V) コネクタに4kΩ～8kΩの抵抗性負荷を適用します。
5. LV (12V) コネクタに適切な抵抗性負荷を適用します。

注: TIでは、絶縁型DC電源を使用して、400V DCおよび12V入力を供給することを推奨します。

6. 400V DC電源を約225V DC出力に設定します。
7. この電源をオンにします。
8. 12V DC電源を12V出力に設定します。
9. この電源をオンにします。
10. start_converterを1に設定します。

注: これにより、降圧モードでDC-DC変換が開始されます。Gui_VlvSetコマンドは、start_converterを1に設定する前のLV電源出力に基づき、LV電源出力よりも約0.5V高く設定されます。

図 69 は、HV電圧が約223V、10Ω負荷で出力が12Vのときの降圧モードのシステム動作に対応する監視ビューを示しています。HVコネクタには8kΩの負荷も接続されています。

Expression	Type	Value	Address
(x)= Gui_Vhv	int	223.344 (Q-Value(5))	0x0000043E@Data
(x)= Gui_VhvSet	int	227.625 (Q-Value(5))	0x00000421@Data
(x)= Gui_Vlv	int	12.1172 (Q-Value(10))	0x0000043A@Data
(x)= Gui_VlvSet	int	12.1152 (Q-Value(10))	0x00000420@Data
(x)= Gui_Ihv	int	0.229736 (Q-Value(12))	0x0000043B@Data
(x)= Gui_Il原因	int	0.6875 (Q-Value(8))	0x00000427@Data
(x)= Buck0_Boost1	int	0	0x00000431@Data
(x)= start_converter	int	1	0x00000436@Data
(x)= Boost_ON	int	0	0x0000043D@Data
(x)= Buck_ON	int	1	0x00000435@Data
(x)= phase	long	0.6313103437 (Q-Value(24))	0x00000472@Data
(x)= BoostDuty	long	0.0 (Q-Value(24))	0x0000046C@Data
(x)= OC_FaultFlg	int	0	0x0000040B@Data
(x)= HVbus_OV	int	0	0x00000432@Data
(x)= LVbus_OV	int	0	0x0000040A@Data
(x)= HVbus_UV	int	0	0x00000430@Data
(x)= LVbus_UV	int	0	0x00000433@Data
(x)= Ihv_trip	int	0.949982 (Q-Value(15))	0x0000041D@Data
(x)= Il原因_trip	int	0.949982 (Q-Value(15))	0x0000041C@Data
(x)= dbA1	int	32	0x00000410@Data
(x)= dbB2	int	18	0x00000413@Data
(x)= dbAtoP_leg	int	32	0x00000414@Data
(x)= dbPtoA_leg	int	18	0x00000415@Data
(x)= auto_DB	int	1	0x00000411@Data
(x)= SR_mode	int	2	0x00000417@Data
+ Add new expression			

図 69. 完全なプロジェクトの“Expressions”ウィンドウ(ランタイム) – 降圧動作

LV出力電圧設定コマンド(Gui_VlvSet)を変更できます。この値は、LV電源設定より低くは設定しないでください。

注: 降圧モードで動作中は、Gui_VlvSetコマンドを12V DC電源電圧設定よりも低く設定してはなりません。

11. Buck0_Boost1を1に設定して、動作を降圧モードから昇圧モードに切り替えます。(Gui_VhvSetコマンドは、Buck0_Boost1を1に設定する前のHV電源出力に基づき、HV電源出力よりも約4V高く設定されます。)

図 70 は、HV電圧出力が約227V、8k Ω 負荷で入力11.6Vのときの昇圧モードのシステム動作に対応する監視ビューを示しています。LVコネクタには100 Ω の負荷も接続されています。

Expression	Type	Value	Address
(x)- Gui_Vhv	int	227.188 (Q-Value(5))	0x0000043E@Data
(x)- Gui_VhvSet	int	227.219 (Q-Value(5))	0x00000421@Data
(x)- Gui_Vlv	int	11.6123 (Q-Value(10))	0x0000043A@Data
(x)- Gui_VlvSet	int	12.1152 (Q-Value(10))	0x00000420@Data
(x)- Gui_Ihv	int	0.546631 (Q-Value(12))	0x0000043B@Data
(x)- Gui_Ilv	int	2.48438 (Q-Value(8))	0x00000427@Data
(x)- Buck0_BoostL	int	1	0x00000431@Data
(x)- start_converter	int	1	0x00000436@Data
(x)- Boost_ON	int	1	0x0000043D@Data
(x)- Buck_ON	int	0	0x00000435@Data
(x)- phase	long	0.003999948502 (Q-Value(24))	0x00000472@Data
(x)- BoostDuty	long	0.3077166677 (Q-Value(24))	0x0000046C@Data
(x)- OC_FaultFlg	int	0	0x0000040B@Data
(x)- HVbus_OV	int	0	0x00000432@Data
(x)- LVbus_OV	int	0	0x0000040A@Data
(x)- HVbus_UV	int	0	0x00000430@Data
(x)- LVbus_UV	int	0	0x00000433@Data
(x)- Ihv_trip	int	0.949982 (Q-Value(15))	0x0000041D@Data
(x)- Ilv_trip	int	0.949982 (Q-Value(15))	0x0000041C@Data
(x)- dbA1	int	32	0x00000410@Data
(x)- dbB2	int	18	0x00000413@Data
(x)- dbAtoP_leg	int	32	0x00000414@Data
(x)- dbPtoA_leg	int	18	0x00000415@Data
(x)- auto_DB	int	1	0x00000411@Data
(x)- SR_mode	int	2	0x00000417@Data
+ Add new expression			

図 70. 完全なプロジェクトの“Expressions”ウィンドウ(ランタイム) – 昇圧動作

注: HV出力電圧設定コマンド (Gui_VhvSet)を変更できます。この値は、HV電源設定より低くは設定しないでください。


注: 昇圧モードで動作中は、Gui_VhvSetコマンドを400V DC電源電圧設定よりも低く設定してはなりません。

常に出力電圧を慎重に観測してください。電圧が基板の定格を超えないようにしてください。

オシロスコープを使用して、PWMゲート駆動信号、入力電圧、電流、出力電圧などの波形をプローブできます。

この絶縁型DC-DCコンバータに対してこれらの高電圧および高電流をプローブする際には、安全性に関する適切な予防措置を実施し、適切な接地要件を考慮してください。

リアルタイム・モードでマイコンを停止させるには、次の手順に従います。

1. 400V DC入力をオフにし、数秒間待ちます。
2. ツールバーの“Halt”をクリックして、プロセッサを停止します。
3.  をクリックして、マイコンのリアルタイム・モードを終了します。
4. マイコンをリセットします。

12 関連資料

1. *UCC28950 600-W, Phase-Shifted, Full-Bridge Application Report* ([SLUA560B](#))
2. *600-W, Phase-Shifted, Full-Bridge Converter* ([SLUU421A](#))
3. Nene, H, “Implementing Advanced Control Strategies for Phase Shifted Full-Bridge DC-DC Converters using Micro-Controllers” *PCIM Europe 2011, Nuremberg, Germany*.
4. Nene, H, “Digital control of a Bi-directional DC-DC Converter for Automotive Applications”, in *Applied Power Electronics Conference and Exposition, (APEC) 2013, pp. 1360–1365*
5. *HV Phase Shifted Full bridge Developer’s Kit* ([TMDSHVPSFBKIT](#))
6. **Bi_dir_Calculations.xls** – このプロジェクトの主要な計算を示すスプレッドシート: ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\v1_00_00_00\Docs
7. **HWdevPkg** – この基板のハードウェア開発に関連した各種のファイルを含むフォルダ: ..\controlSUITE\development_kits\BI_DIRECTIONAL_DC_DC_400_12\v1_00_00_00\HWdevPkg

12.1 商標

C2000, Code Composer Studio are trademarks of Texas Instruments.

Mathcad is a registered trademark of PTC Inc.

MATLAB is a registered trademark of The MathWorks, Inc.

すべての商標および登録商標はそれぞれの所有者に帰属します。

13 著者について

HRISHIKESH NENEは、2006年からTIのC2000システム・チームに属しています。デジタル制御のパワー・エレクトロニクス・ベースのシステムに関して、システムおよびリファレンス・ソリューションの設計・開発に幅広く取り組んできました。ここ数年間は、主に絶縁型および非絶縁型DC-DCコンバータの制御に焦点を合わせて活動しています。

TIの設計情報およびリソースに関する重要な注意事項

Texas Instruments Incorporated ("TI")の技術、アプリケーションその他設計に関する助言、サービスまたは情報は、TI製品を組み込んだアプリケーションを開発する設計者に役立つことを目的として提供されるものです。これにはリファレンス設計や、評価モジュールに関係する資料が含まれますが、これらに限られません。以下、これらを総称して「TIリソース」と呼びます。いかなる方法であっても、TIリソースのいずれかをダウンロード、アクセス、または使用した場合、お客様(個人、または会社を代表している場合にはお客様の会社)は、これらのリソースをここに記載された目的のみに使用し、この注意事項の条項に従うことに合意したものとします。

TIによるTIリソースの提供は、TI製品に対する該当の発行済み保証事項または免責事項を拡張またはいかなる形でも変更するものではなく、これらのTIリソースを提供することによって、TIにはいかなる追加義務も責任も発生しないものとします。TIは、自社のTIリソースに訂正、拡張、改良、およびその他の変更を加える権利を留保します。

お客様は、自らのアプリケーションの設計において、ご自身が独自に分析、評価、判断を行う責任がお客様にあり、お客様のアプリケーション(および、お客様のアプリケーションに使用されるすべてのTI製品)の安全性、および該当するすべての規制、法、その他適用される要件への遵守を保証するすべての責任をお客様のみが負うことを理解し、合意するものとします。お客様は、自身のアプリケーションに関して、(1) 故障による危険な結果を予測し、(2) 障害とその結果を監視し、および、(3) 損害を引き起こす障害の可能性を減らし、適切な対策を行う目的での、安全策を開発し実装するために必要な、すべての技術を保持していることを表明するものとします。お客様は、TI製品を含むアプリケーションを使用または配布する前に、それらのアプリケーション、およびアプリケーションに使用されているTI製品の機能性を完全にテストすることに合意するものとします。TIは、特定のTIリソース用に発行されたドキュメントで明示的に記載されているもの以外のテストを実行していません。

お客様は、個別のTIリソースにつき、当該TIリソースに記載されているTI製品を含むアプリケーションの開発に関連する目的でのみ、使用、コピー、変更することが許可されています。明示的または黙示的を問わず、禁反言の法理その他のような理由でも、他のTIの知的所有権に対するその他のライセンスは付与されません。また、TIまたは他のいかなる第三者のテクノロジーまたは知的所有権についても、いかなるライセンスも付与されるものではありません。付与されないものには、TI製品またはサービスが使用される組み合わせ、機械、プロセスに関連する特許権、著作権、回路配置利用権、その他の知的所有権が含まれますが、これらに限られません。第三者の製品やサービスに関する、またはそれらを参照する情報は、そのような製品またはサービスを利用するライセンスを構成するものではなく、それらに対する保証または推奨を意味するものでもありません。TIリソースを使用するため、第三者の特許または他の知的所有権に基づく第三者からのライセンス、あるいはTIの特許または他の知的所有権に基づくTIからのライセンスが必要な場合があります。

TIのリソースは、それに含まれるあらゆる欠陥も含めて、「現状のまま」提供されます。TIは、TIリソースまたはその仕様に関して、明示的か暗黙的にかかわらず、他のいかなる保証または表明も行いません。これには、正確性または完全性、権原、続発性の障害に関する保証、および商品性、特定目的への適合性、第三者の知的所有権の非侵害に対する黙示の保証が含まれますが、これらに限られません。

TIは、いかなる苦情に対しても、お客様への弁済または補償を行う義務はなく、行わないものとします。これには、任意の製品の組み合わせに関連する、またはそれらに基づく侵害の請求も含まれますが、これらに限られず、またその事実についてTIリソースまたは他の場所に記載されているか否かを問わないものとします。いかなる場合も、TIリソースまたはその使用に関連して、またはそれらにより発生した、実際の、直接的、特別、付随的、間接的、懲罰的、偶発的、または、結果的な損害について、そのような損害の可能性についてTIが知らされていたかどうかにかかわらず、TIは責任を負わないものとします。

お客様は、この注意事項の条件および条項に従わなかったために発生した、いかなる損害、コスト、損失、責任からも、TIおよびその代表者を完全に免責するものとします。

この注意事項はTIリソースに適用されます。特定の種類の資料、TI製品、およびサービスの使用および購入については、追加条項が適用されます。これには、半導体製品(<http://www.ti.com/sc/docs/stdterms.htm>)、評価モジュール、およびサンプル(<http://www.ti.com/sc/docs/sampterm.htm>)についてのTIの標準条項が含まれますが、これらに限られません。