位相シフト・フルブリッジ CCS ユーザー・ガイド

Version 1.1
2012年3月

C2000マイクロコントローラを使用した 位相シフト・フルブリッジ DC - DCコンバータの実装

Hrishikesh Nene
C2000 Systems and Applications Team

概要

位相シフト・フルブリッジ(PSFB)・コンバータは、各種アプリケーションでのDC-DC変換に使用されます。例えば電気通信システムでは、高電圧バスを中間配電電圧(通常は約48V)に変換するために使用されます。このトポロジーにはトランス(変圧回路)も含まれているため、PSFB段を使用すると、電圧変換とライン電圧からの絶縁分離(isolation)の両方が実現されます。

このドキュメントでは、Texas InstrumentsのHVPSFBキットに実装されたデジタル制御PSFBシステムの実装の詳細を紹介します。このキットではDC入力400Vを安定化DC出力12Vに変換し、最大600Wで動作します。ピーク電流モード制御(PCMC)と電圧モード制御(VMC)の両方の実装が説明されます。これらの高度に統合された、マイクロコントローラ・ベースの実現は、適応型ゼロ電圧スイッチング(ZVS)・スキームと各種同期整流スキームを特徴としています。このドキュメントでは、これらの機能について説明します。また、これらの制御スキームで必要となる複雑なゲート・ドライブ波形の生成の詳細と、動作条件の変化に合わせてシステム性能を最適化するためのインテリジェントなタイミング制御も紹介されます。HVPSFBプロジェクトを実行し、慣れるためのステップ分けされたガイドも記載されています。10%超の定格を持つ高システム効率で安定した負荷、オンチップ・ハードウェア機構をベースとした、革新的なPCMC波形生成、シンプルなシステム実装が、このソリューションの特長です。

目次

図目次

表目次

注: 実装の各手順をある程度省略してこのキットを短時間で評価したい場合は、このドキュメントではなく、同梱のクイック・スタート・ガイド(QSG-HVPSB-Rev1.1.pdf)を読んでください。

1 はじめに

位相シフト・フルブリッジ(PSFB)DC-DCコンバータは、高いDCバス電圧の降圧、および/または中～高電力アプリケーション(サーバー電源、通信用整流器、充電システム、再生可能エネルギー・システム等)での絶縁分離(isolation)の実現によく使用されます。従来は、上記システム内でのマイクロコントローラの働きは、監視タスクや通信タスクのみに限られていました。現在では高性能のマイクロコントローラ・デバイスが入手可能になったため、従来のマイクロコントローラ機能の利用に加えて、上記システム内の制御閉ループにマイクロコントローラを使用することも可能になりました。デジタル出力制御への移行は、従来はハードウェア的に実装されていた機能がソフトウェア的に実装されるようになったことを意味します。これにより、システムに柔軟性が付加されるばかりでなく、システムが大幅に簡素化されます。このようなシステムでは、様々な条件下で電力段を最適に制御するとともに、システムレベルのインテリジェンスを実現する高度な制御手法の実装が可能になります。

PSFBコンバータは、(MOSFETやIGBT等の)4つのパワー・エレクトロニック・スイッチで構成されます。この4つのスイッチは、絶縁トランスの1次側でフルブリッジ回路、2次側で同期整流(SR)用のダイオード整流器またはMOSFETスイッチを形成しています。このトポロジーではすべてのスイッチング・デバイスがゼロ電圧スイッチング(ZVS)でスイッチングを行えるため、スイッチング損失が低減して、コンバータの効率が向上します。この例では、負荷の状態に応じて1次側のスイッチのデッド・タイム(不感時間)を変化させることにより、完全な負荷範囲全体で、フルブリッジ回路の一方のレッグのスイッチでゼロ電圧スイッチング(ZVS)が実現され、もう一方のレッグのスイッチでゼロまたは低電圧スイッチング(バレー・スイッチング)が実現されます。

上記のような絶縁型トポロジーでは、2次側で信号の整流(signal rectification)が必須になります。低出力電圧定格および/または高出力電流定格のシステムの場合は、ダイオード整流でなく同期整流を実装することでダイオード整流の損失が回避できるため、可能な範囲で最高の性能が実現されます。この例では各種の(different)スイッチング・スキームを使用して倍電流同期整流(current doubler synchronous rectification)を2次側に実装することにより、様々な負荷条件下で最適の性能を実現します。

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)、ピーク電流モード制御(PCMC)等の各種モードで制御できます。通常であれば、同一電力段の制御用に上記の制御モードを実装するには、電力段のセンシング回路にある程度変更を加えるだけでなく制御回路を設計し直すことが必要になります。ところがマイクロコントローラ・ベースのシステムを使用している場合は、最小限の変更のみで、あるいはまったく変更を加えずに、同一の設計で上記のすべてのモードを試すことが可能です。このアプリケーション・ノートでは、VMCおよびPCMC制御スキームを使用してこのようなシステムを実装します。

PCMCには内蔵電圧フィードフォワード機能、サイクル単位の自動電流制限機能、磁束バランス(flux balancing)機能等の利点があり、パワー・コンバータ(電力変換回路)に非常に適した制御スキームです。PSFBシステム用にPCMCを実装するには、高精度のタイミング制御を使用した複雑なPWM波形生成が必要になります。マイクロコントローラ製品TMS320F2802xとTMS320F2803x (Texas InstrumentsのPiccoloシリーズ)を使用すると、補助回路を一切追加することなく、上記の波形生成を斬新なアプローチで実現することが可能になります。プログラム可能な独自のオンチップ・スロープ補償ハードウェアを使用して適切なスロープ補償を行うことで、開ループの安定性を保証し、出力での分周波(低調波)の発振を除去/制限します。マイクロコントローラにPCMCを実装する場合は、安定化された出力電圧が出力電圧リプルの量に依存します。出力電圧リプル自体は負荷に依存します。この関係を詳細に説明し、各種のソリューションを提案します。

このドキュメントで使用される600WのPSFBシステムでは、95%を超えるピーク効率と、負荷10%に対する90%を超える効率が実現されます。

1.1 基本的な動作

図1は、位相シフト・フルブリッジ回路の簡略図です。MOSFETスイッチQ_A、Q_B、Q_C、Q_Dにより、トランスT1の1次側でフルブリッジ回路が形成されています。Q_AとQ_Bはデューティ比50%および互いの位相ずれ180度でスイッチングされます。同様に、Q_CとQ_Dもデューティ比50%および互いの位相ずれ180度でスイッチングされます。フルブリッジ回路のレッグQ_C～Q_DのPWMスイッチング信号は、レッグQ_A～Q_BのPWMスイッチング信号に対して位相シフトされます。この位相シフトの量により、対角線上に位置するスイッチ間の重複(overap)の量が決まります。そして、この重複の量により、伝達されるエネルギーの量が決まります。D_1、D_2により2次側のダイオードの倍電流整流(diode current doubler rectification)が実現し、LoとCoにより出力フィルタが形成されます。インダクタL_Rが、トランスの漏れインダク

タンスとMOSFET容量による共振動作(resonance operation)を補助して、ゼロ電圧スイッチング(ZVS)が促進されます。トランスT1の両側に1つずつあるグラウンドG1とG2に注目してください。図2に、図1のシステムのスイッチング波形を示します。

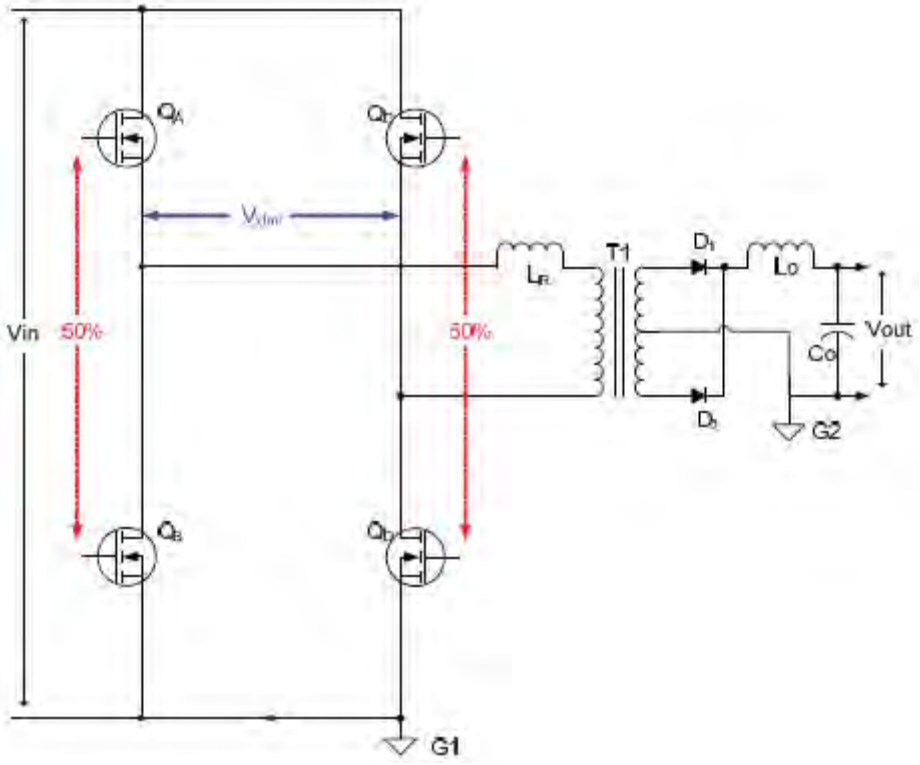


図 1 位相シフト・フルブリッジ 回路

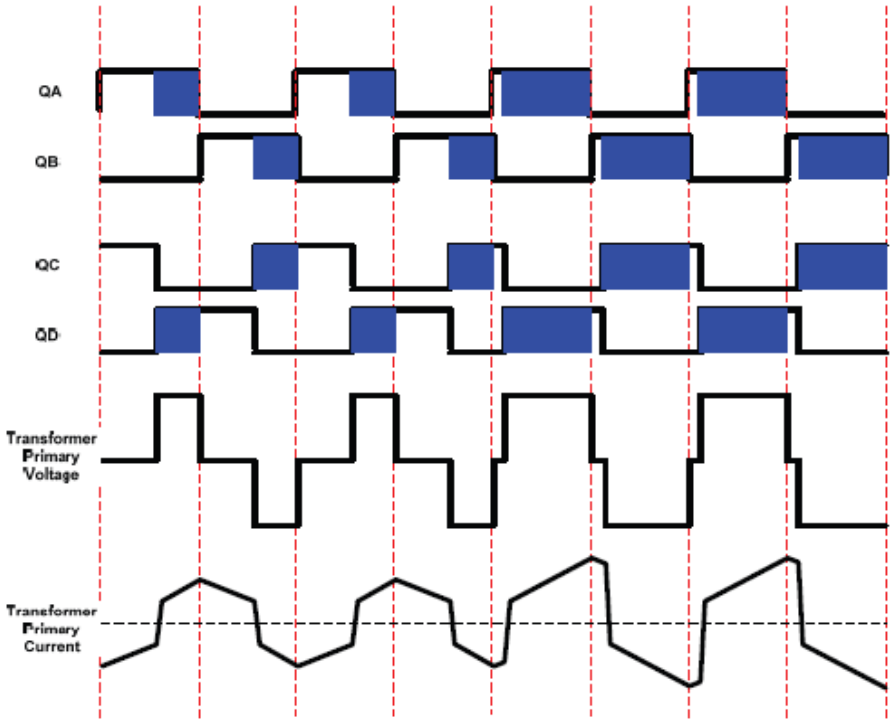


図 2 PSFB PWM 波形

1.2 HVPSFB ボードへの PSFB の実装

図3は、HVPSFBボードに実装されたPSFB回路の簡略ブロック図です。図2のスイッチQ_A、Q_B、Q_C、Q_Dは、スイッチQ1、Q4、Q2、Q3にそれぞれ対応しています。スイッチQ5とQ6は、2次側の同期整流に使用されます。

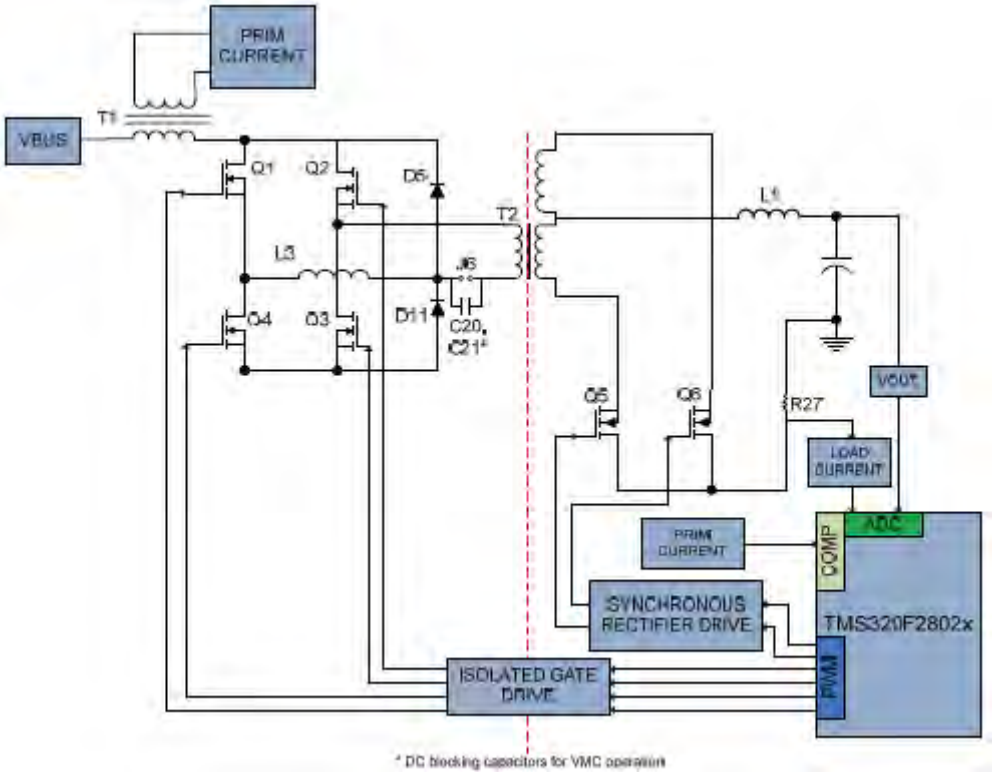


図 3 PSFB システムのブロック図

制御アルゴリズムは、C2000マイクロコントローラ(MCU)に実装されています。MCUは、フィードバック信号とPWM出力を手段としてPSFBの電力段と通信(相互作用)を行います。コントローラは、この設計では2次側に配置されています。絶縁境界に関連してコントローラの配置を決めることは、絶縁型DC-DCシステムを設計する際のきわめて重要なステップです。複数の出力レールを備えていたり、2次側で多くの信号と制御ループを処理したり、(2次側の)アプリケーションの他のシステムと通信を行ったりするシステムでは、コントローラを2次側に配置することでより多くの利点が得られます。

フルブリッジ回路の2つのレッグを駆動するPWM信号間の位相シフトにより、負荷に伝達されるエネルギーの量が決まります。この位相シフトは、制御されたパラメータです。

MCUでは、この位相シフトを制御して出力電圧を安定化し、指令した値に保持することにより、DC–DC変換を実現します。

各種動作モードでこのようなシステムを制御するには、高速で高効率の制御ループ計算に加えて、複雑なPWM駆動波形の生成が必要になります。これは、、高効率の32ビットCPUに結合されているPWMモジュール、DAC付きのアナログ・コンパレータ、、スロープ補償ハードウェア、12ビット高速ADC等の高度なオンチップ制御ペリフェラルを併せて使用することで、C2000マイクロコントローラによって実現可能になります。以降の各章では、ソフトウェア・アルゴリズムの詳細について説明します。

1.3 HVPSFB キットの特長

HVPSFBキットの主な機能を次に挙げます。

- DC入力400V (370Vdc～410Vdc動作)、DC出力12V
- 95%を超えるピーク効率。負荷10%に対する90% を超える効率
- 定格50アンペア(600ワット)の出力

- 位相シフト・フルブリッジ 回路トポロジー
- 100KHzのスイッチング周波数
- PCMC機能用のピーク電流モード制御(PCMC) (外部補助回路なし)
- 複数の同期整流(SR)スイッチング・スキーム
- 全負荷範囲での適応型ZVS/LVS
- 手早く簡単にシステム調整(system tuning)を実施し、最適な性能を実現できる、高効率のGUIインターフェイス
- 故障保護: 入力UVとOV、過電流、出力UV (CC/CPモード)
- 定電流(CC)機能と定電力(CP)機能
- オプションとして電圧モード制御(VMC: Voltage Mode Control)

次に、このキットを使用して得られた結果の例をいくつか紹介します。

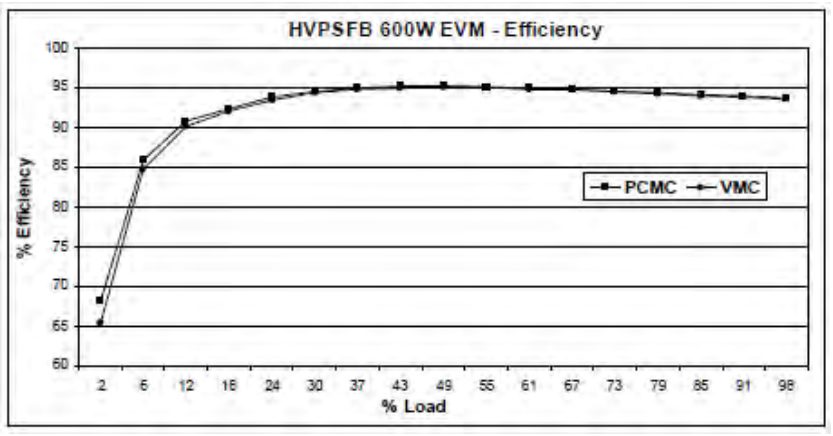


図 4 効率対負荷(PCMC実装とVMC実装)

* 補助電力は、効率の計算に含まれていないことに注意してください。

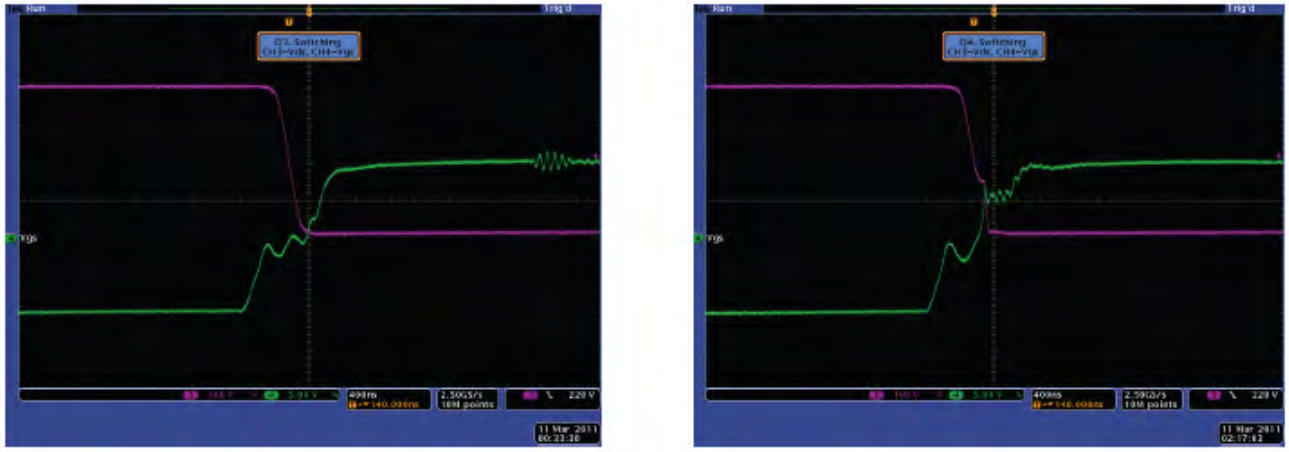
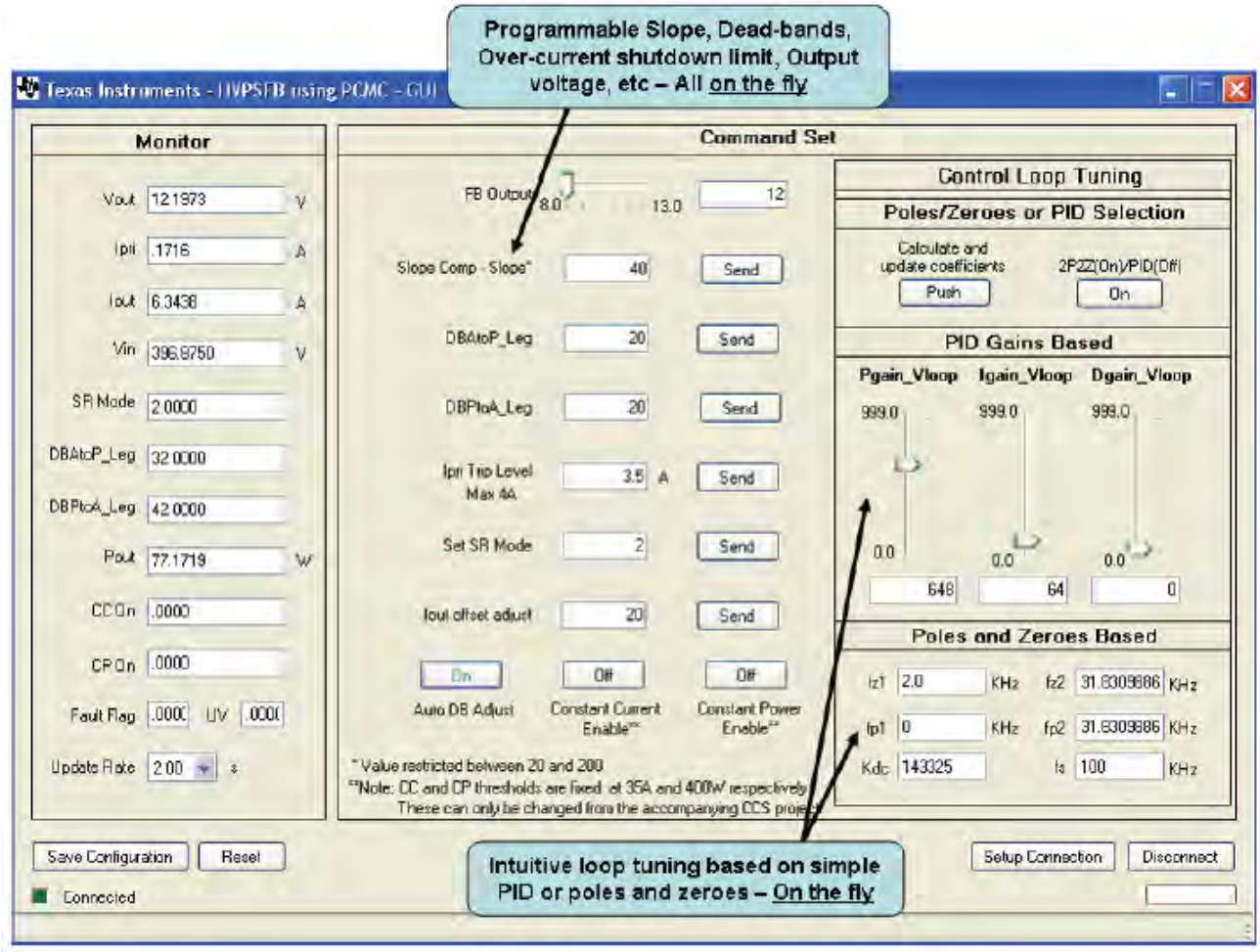


図 5 負荷12Aの場合のZVS/LVS スイッチング (a) 能動レッグから受動レッグへの遷移(ZVS) (b) 受動レッグから能動レッグへの遷移 (LVS)

図 6 PCMCの過渡応答 (a) 0%から80%への負荷ステップ(load step) (b) 80%から0%への負荷ステップ

ZVSおよび/またはLVSスイッチングは、完全な動作範囲全体で実現されます。定格10%よりも大きいすべての負荷では90%を超えるシステム効率が得られる一方で、ピーク効率が95%よりも大きくなります。負荷のステップ変化が80%の場合は、出力最大偏差(peak deviations)は定格の3%以下、セトリング時間250us以内が実現されます。PCMCのステップ応答(step response)は、減衰した1次システム(damped first order system)に似ています。C2000マイコン(TMS320F2802x)をベースとした実装では、PCMCおよびVMC制御スキームに必要な複合型ゲート・ドライブ波形の生成・制御機能が実現される一方で、デジタル制御されたソリューションならではの一定レベルのインテリジェンスが保たれます。

このキットに付随のソフトウェア・パッケージに含まれているPCMC GUIのスナップショット(画面例)を次に示します。VMC実装用のGUIも同梱されています。



1.4 ボードの主要な部品の説明

ここで、実際のハードウェア上の主な部品をいくつか確認しておきます。

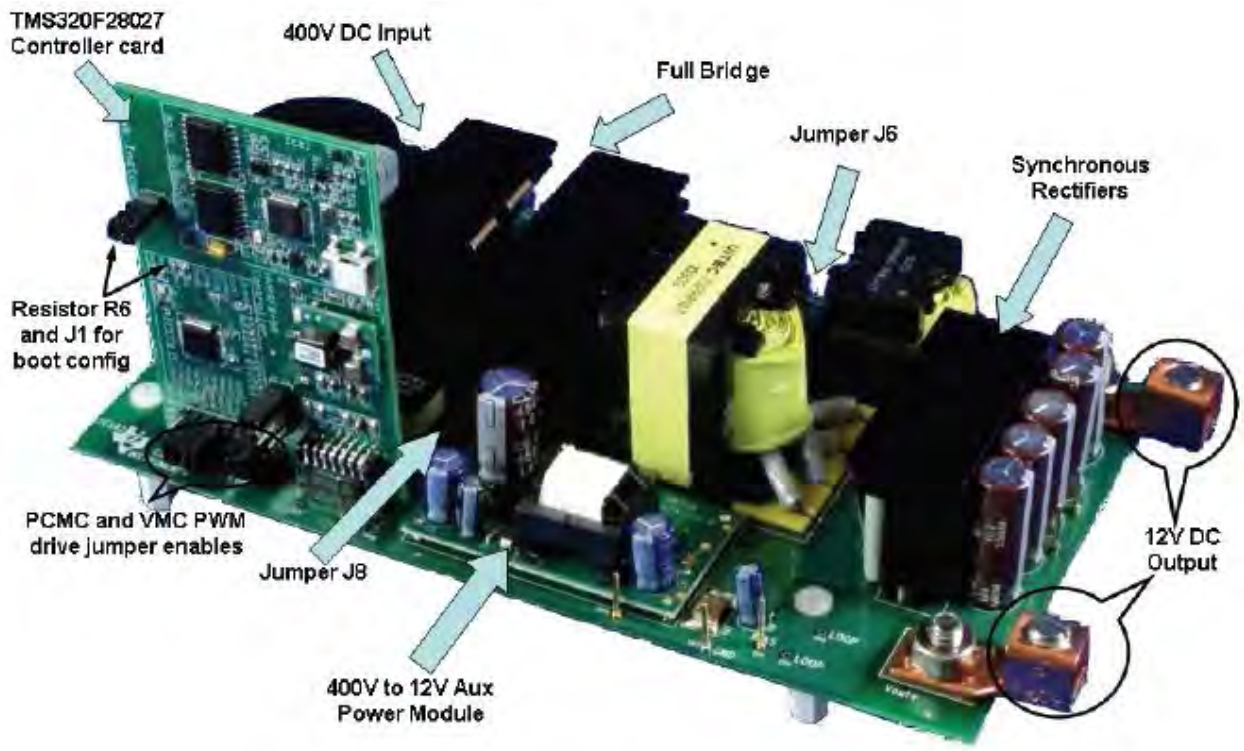


図 8 HVPSFB ボードとコントローラ・カード

図9と図10は、Piccolo-A コントローラ・カードとHVPSFBベース・ボードの 回路図です。

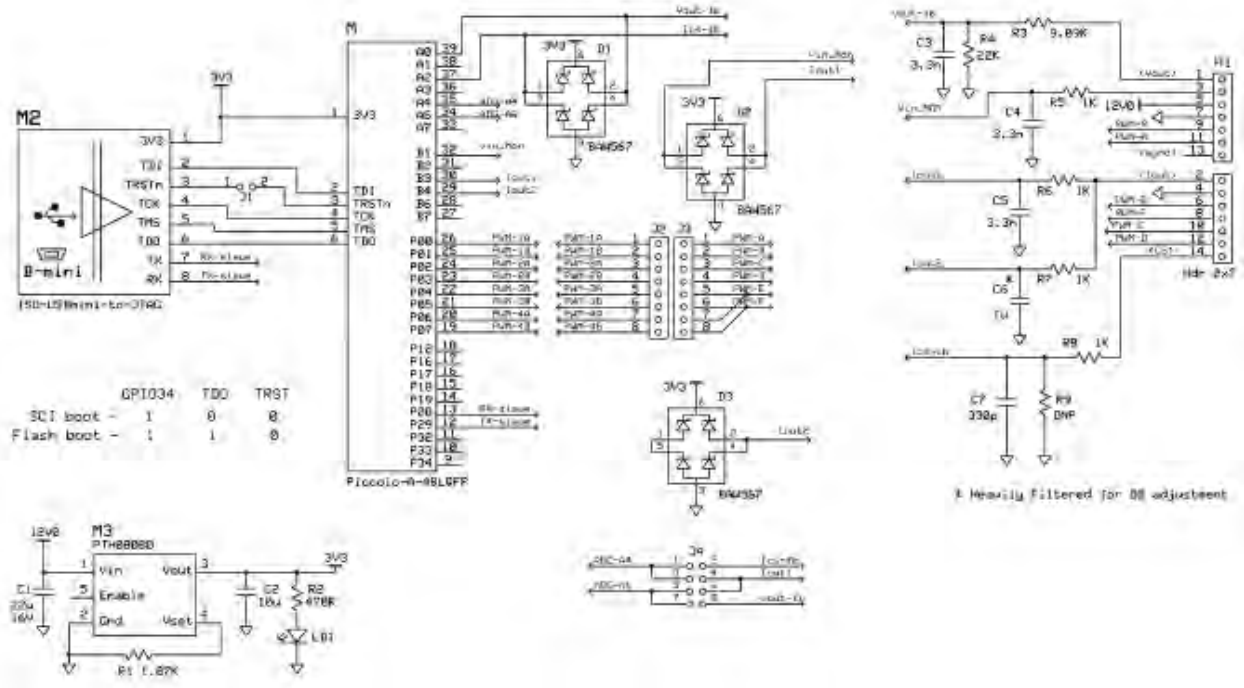


図 9 TMS320F28027(Piccolo-A) コントローラ・カードの回路図

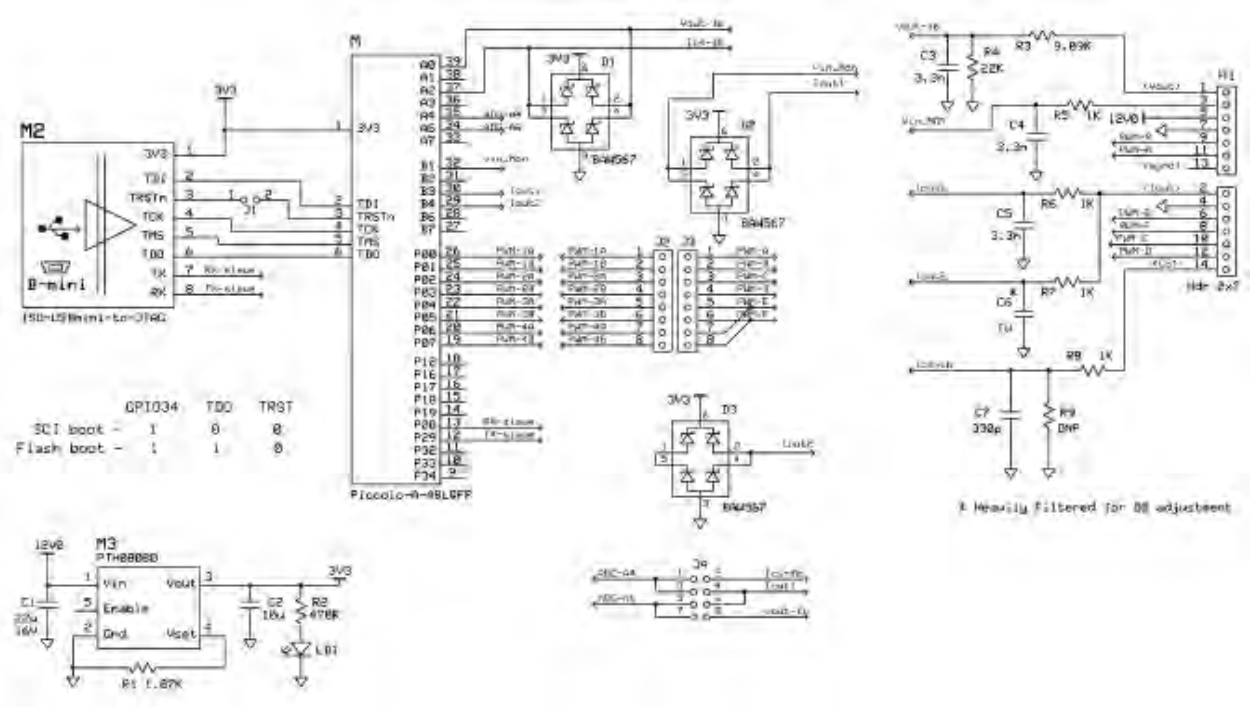


図10 HVPSFBベース・ボードの回路図

2 機能の説明

2.1 ピーク電流モード制御 (PCMC)

PSFBシステム用にPCMCを実装するには、高精度のタイミング制御を使用した複合型のPWM波形生成が必要になります。Texas InstrumentsのPiccoloファミリのデバイスの特長である高度なオンチップ制御ペリフェラルを使用すれば、この目的のために外部補助回路を一切使用せずにこの実装が可能になります。上記のペリフェラルとしては、オンチップ・アナログ・コンパレータ、デジタル-アナログ・コンバータ(DAC)、高度なPWMリソース、独自のプログラム可能オンチップ・スロープ補償ハードウェアなどがあります。図11は、PCMC実装のブロック図です。トランスの1次電流が、電圧ループにより計算されたピーク基準電流と、オンチップ・コンパレータ1を使用して比較されます。図12に示すように、トランスの1次電流が指定された(commanded)ピーク・リファレンス値に達すると、スイッチ(Q2/Q3)を駆動しているPWM波形の一方が1/2スイッチング・サイクルごとに「リセット」され、その直後に電力伝達位相が終了します。同じレッグ内で残りのスイッチを駆動しているPWM波形は、プログラム可能なデッド・タイム(デッド・バンド)・ウィンドウの後で「セット」されます。適切なスロープ補償も施されて、プログラマブルな負のスロープを持つランプ波(ramp)がピーク基準電流信号に付加されます。同一レッグ内の複数PWMの「リセット」動作と「セット」動作の結果、2つのレッグを駆動するPWM信号間で位相シフト(位相のずれ)が発生します。この位相シフトの量と、位相シフトの結果発生する対角線上のスイッチ間の重複(overap)は、ピーク基準電流の量に依存します。ピーク基準電流が大きいほど、対角線上のスイッチ間の重複(overap)が長くなるため、2次側にそれだけ多くのエネルギーが伝達されることになります。コントローラではピーク基準電流値の制御によりこのエネルギー伝達を制御することで、出力を安定化します。したがって、このピーク基準電流は制御されたパラメータになります。

この実装の重要な特徴のひとつは、あらゆる動作条件下で、スイッチング・サイクルの両方の側で同じピーク基準電流コマンドが使用されることです。これによりトランスの1次側で最適な磁束バランスが実現され、どのような場合でも飽和の可能性が低くなります。

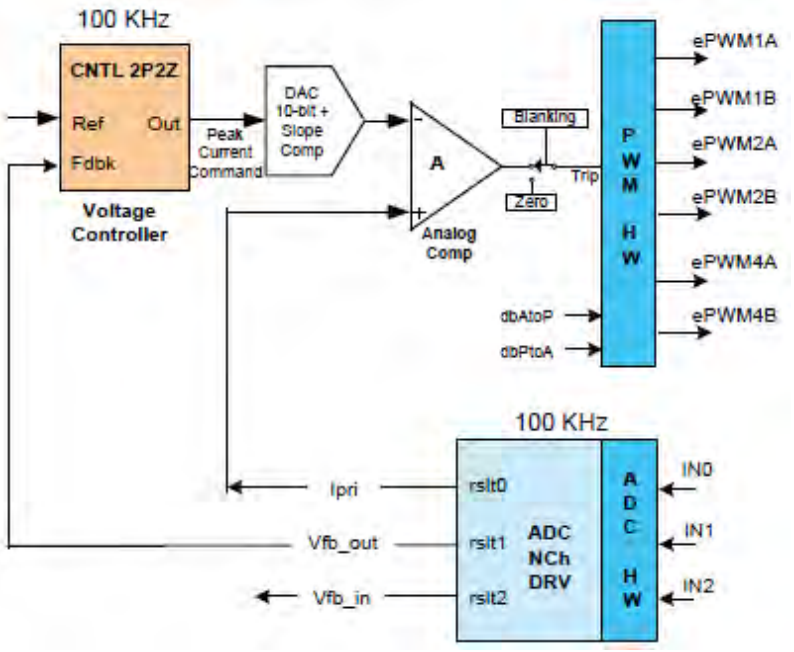


図11 PCMCのブロック図

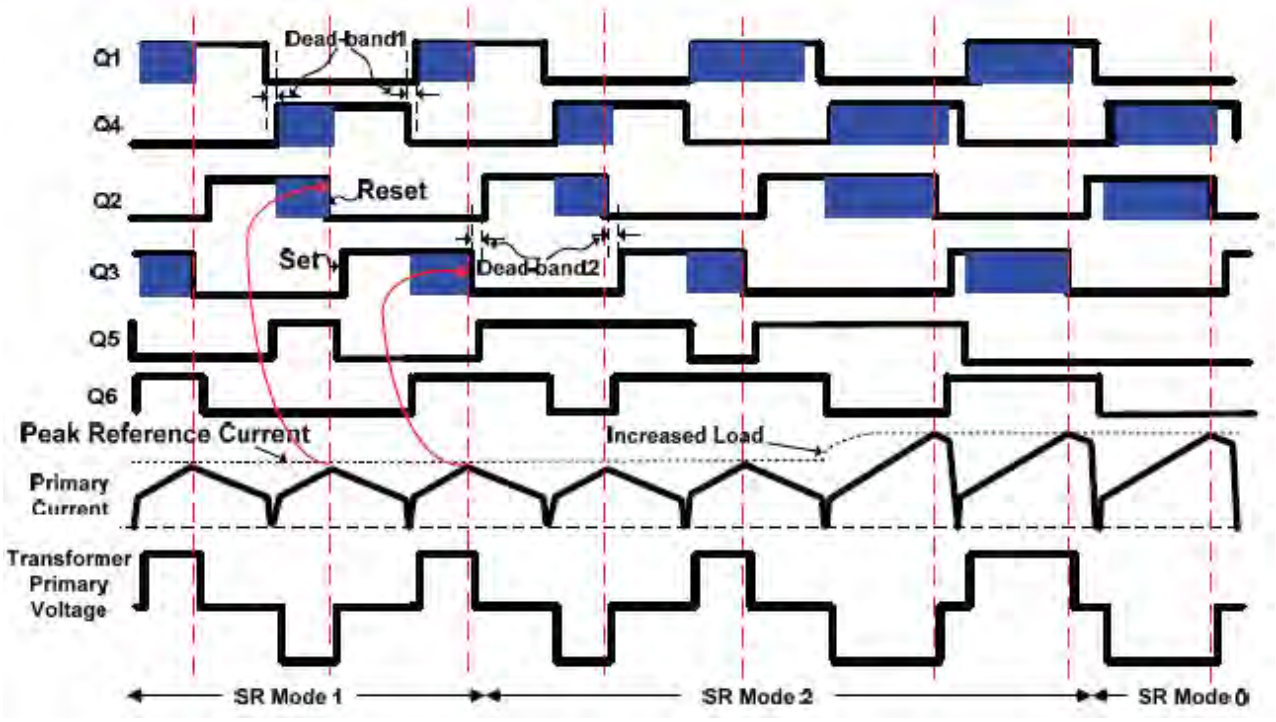


図12 PCMCのPWM波形

2.2 電圧モード制御(VMC)

VMCの実装では、固定(50%)デューティ・サイクルと周波数の相補型パルス幅変調(PWM)信号を使用して、各レッグのスイッチが駆動されます。図13のように、コントローラではブリッジの一方のレッグのスイッチを駆動しているPWM信号に関連して、他方のレッグのスイッチを駆動しているPWM信号の位相シフトの駆動と制御を直接行います。この位相シフトにより、対角線上で向き合うスイッチ間の重複(overap)の量が検出されます。これは図14を見るとよく分かります。対角線上のスイッチ間の重複(overap)が長いほど、トランスの1次巻き線の両端に入力電圧がかかる時間が長くなるため、2次側に伝達されるエネルギーの量も大きくなります。コントローラでは、2つのレッグを駆動するPWM信号間の位相シフトの制御によりこのエネルギー伝達を制御することで、出力を安定化します。したがって、この位相シフトは制御されたパラメータになります。VMCを実装する場合には、トランスの1次側にDCブロック・コンデンサを付けて、時間の経過に伴う流速の不均衡化(flux imbalance)によりトランスが飽和しないようにする必要があることに注意してください。したがって、VMC実装の場合には図8のジャンパJ6を外す必要があります。

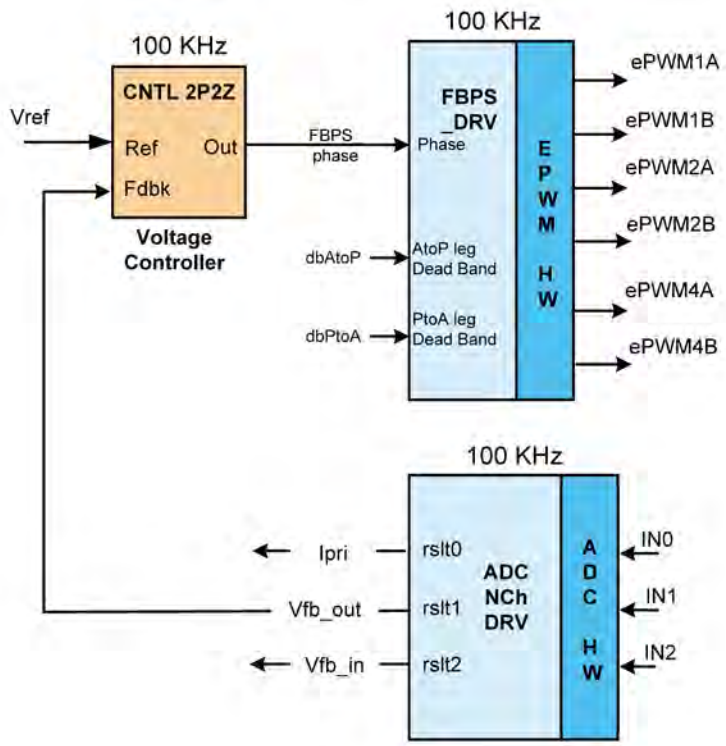


図13 VMCのブロック図

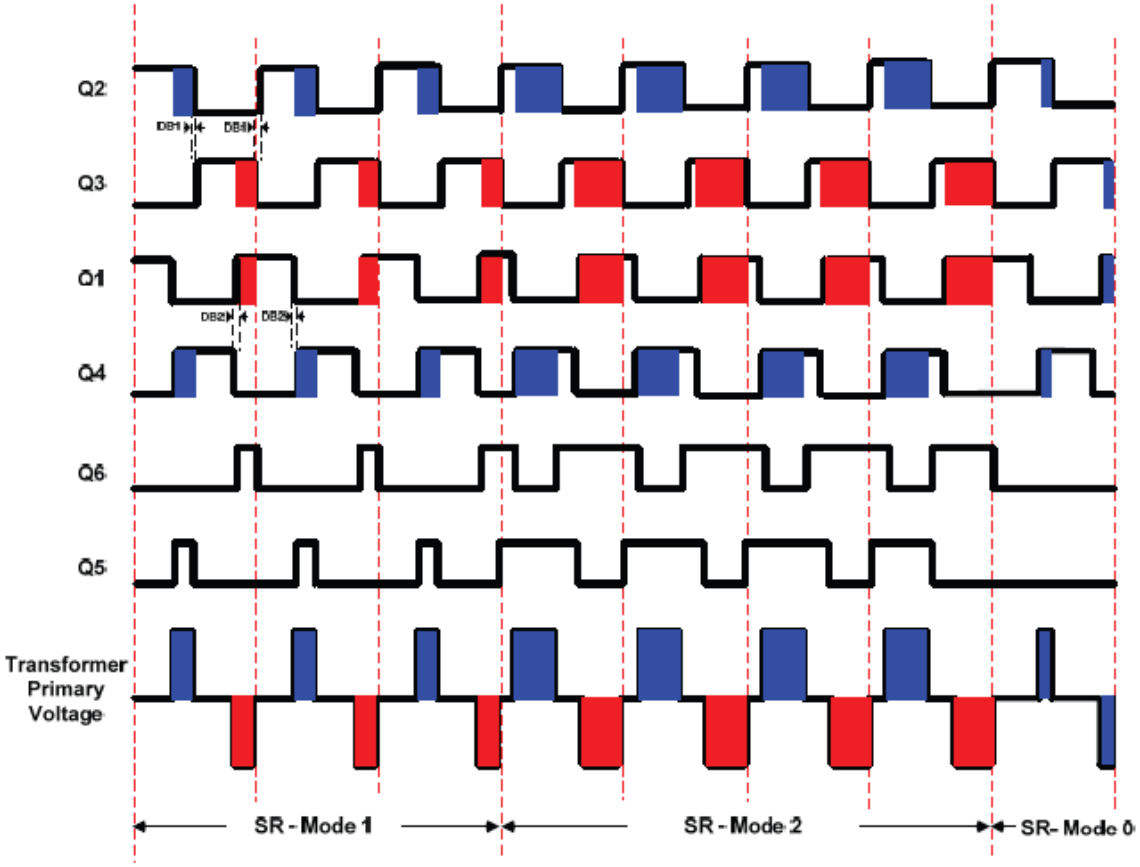


図14 VMCの波形

2.3 ゼロ電圧スイッチング (ZVS)または低電圧スイッチング (LVS)

PSFB DC-DCコンバータでは、回路内の寄生素子を利用して、MOSFETをONにする前にMOSFETスイッチ両端の電圧が確実にゼロになるようにすることで、ソフト・スイッチングを実現しています。これにより、ハード・スイッチングに関連するスイッチング損失の量がかなり小さくなります。

ここで論じるシステムの場合は、Q2～Q3レッグのスイッチのスイッチング遷移(switching transitions) により、電力伝達インターバル(power transfer interval)が終了します。したがってこのレッグは、「能動 – 受動(Active to Passive)」レッグと呼ばれます。 このレッグのスイッチで遷移(transitions)が発生すると、1次巻き線の電流がその側(half)のPWMスイッチング・サイクルの最大振幅(maximum magnitude)に近づきます。反映された(reflected)負荷電流により、この時間に1次側の回路で循環しているエネルギーにさらにエネルギーが加わります。このことで、このレッグのスイッチ両端の電圧が0ボルトに近づくことが可能になります。完全な負荷範囲全体で、このQ2～Q3レッグのスイッチでZVSを実現することが可能です。負荷が減少するにつれて、デッド・タイムを長くしてZVSを実現する/ZVSに近づける必要があることに注意してください。

Q1～Q4レッグのスイッチのスイッチング遷移により、電力伝達インターバルが開始されます。したがってこのレッグは、「受動–能動(Passive to Active)」レッグと呼ばれます。このスイッチング遷移中に1次側の電流が減少し、ゼロ電流値と交差して方向が変わります。この結果、ZVSに使用可能なエネルギーが減ります。実際には、低負荷条件下の動作では、これらのスイッチがオンになる前にスイッチ両端電圧がゼロにならないようになっています。スイッチ両端の電圧が最小限の時点でこれらのスイッチをオンにすると、スイッチング損失を最小限に抑えることができます。これはバレー・スイッチングまたは低電圧スイッチング(LVS)と呼ばれます。負荷が変化すると、スイッチをオンにしてLVS変化を実現する必要のある時間も変化するため、Q2～Q3レッグのスイッチ同様にデッド・タイムの補正が必要になります。

2.4 同期整流

同期整流器は、任意の時点で、次の3つのモードのどれかで動作可能です。

I. モード0: 古典的なダイオードの倍電流(current doubler)モードであり、同期整流器をオフの状態に保持することで実現します。同期整流によって得られる電力節減よりも同期整流器のスイッチング損失が大きくなるほど低い負荷動作に有効です。

II. モード1: このモードでは、同期整流器のスイッチが理想ダイオードのように振る舞います。非常に低い負荷で動作する場合(通常はバースト・モードが使用されている場合)に有効です。このモードでは、対応する対角線上のブリッジの駆動信号が重複する場合のみ、同期整流器のMOSFETがオンになります。

III. モード2: 上記以外の全ての負荷条件に有効です。このモードでは、対応する対角線上で向き合ったブリッジの駆動信号が重複する場合のみ、同期整流器のMOSFETがオフになります。

図12と図14は、同期整流器のスイッチをこれらのモードで駆動するために生成された波形です。大きな負荷過渡(large load transients)コマンドや急な位相シフト変更コマンド中でもPWMの出力にグリッチや異常(anomalies)が発生しないように、モード遷移(mode transitions)をシームレスに実装して、システムの安全な動作を確実にすることが重要です。

2.5 負荷の変化を利用した出力電圧の安定化

安定化された出力電圧は、負荷条件の変化に影響されます。この問題の原因は、出力電圧にスイッチング周波数の2倍でリプルが出現することです。ピーク・トゥ・ピーク電圧リプルは低負荷時には比較的小さく、負荷が大きくなると大幅に増加します。図15のように、ADC変換がスイッチング・サイクル内の決まった時点でトリガされる場合は、検知されたADC電圧の結果が高負荷では小さくなります(平均電圧が同じ場合)。このことは、次のように示されます。

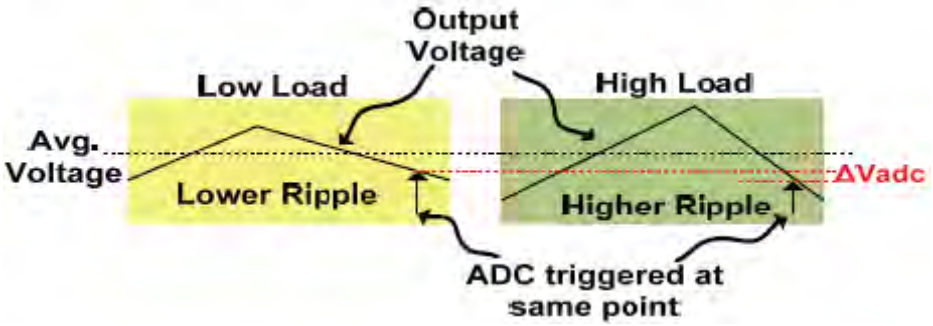


図 15 決まった時点でのADC変換トリガの、出力電圧検知動作への影響

このように、平均電圧が同じであれば、高負荷動作下ではコントローラ側から見た出力電圧がより小さくなります。コントローラではこのΔVadcを補償して、負荷の増大とともに実際の出力電圧値も大きくなるようにします。この影響を最小限に抑えるために可能な解決法(ソリューション)を、次にいくつか紹介します。

1. ADC変換の開始を1/2PWMスイッチング・サイクルごとの適切な時間にトリガすることで、平均出力電圧を直接検知できるようになります。PCMC実装を使用した場合はデューティ比があらかじめ決定されないため、このトリガ・ポイントは未知になります。この方法は、VMCの場合に非常に有効となります。

2. 低速時の(at slow rate)平均出力電圧を計算し、外側のより低速のループを使用して、基準電圧またはフィードバック(電圧)を補正できます。ただし、このことは動的性能(dynamic performance)に影響する可能性があります。

3. 出力電圧を1～2リプル・サイクルにわたってオーバーサンプリングすることで、1サイクルごとの平均出力電圧を計算できます。平均出力電圧は完全な1リプル・サイクルについて計算されるため、より高い/より低いピーク・トゥ・ピーク・リプルからの影響をすべて回避できます。また、平均が1または2リプル・サイクルで計算され、次のPWMスイッチング・サイクルに使用されるため、動的な振る舞いと制御ループの性能(パフォーマンス)があまり劣化しません。この方法を使用すると、単一の1/2PWMサイクル内で複数のADC変換が 必要になります。この方法はPCMC実装に推奨されます。この実装では、1回のPWMスイッチング・サイクル内で出力電圧が8回オーバーサンプリングされます。

4. ADCの入力でのフィルタリング処理を増加させることで、リプルを減衰させ、ΔVadcを小さくすることが可能になります。ただしこの方法では、システムの動的性能と実現可能なループ帯域幅が影響を受けても、出力電圧の振る舞いは変わりません。

3 ソフトウェアの概要 - PCMC

3.1 ソフトウェア制御フロー

HVPSFB_PCMCプロジェクトでは、「C-background/ASM-ISR」フレームワークを利用します。この方法ではアプリケーションの主な支援プログラムとしてCコードを利用し、すべてのシステム管理タスク、意志決定(decision making)、インテリジェンス(intelligence)、ホストとの通信(host interaction)を担当します。アセンブリ・コードは厳密にISR(割り込みサービス・ルーチンInterrupt Service Routine)に限定されます。ISRではすべてのクリティカルな制御コードを実行します。通常、このクリティカルな制御コードにはADC読み取り(reading)、制御計算、PWMとDACのアップデート等が含まれます。図16に、このプロジェクトの一般的なソフトウェア・フローを示します。

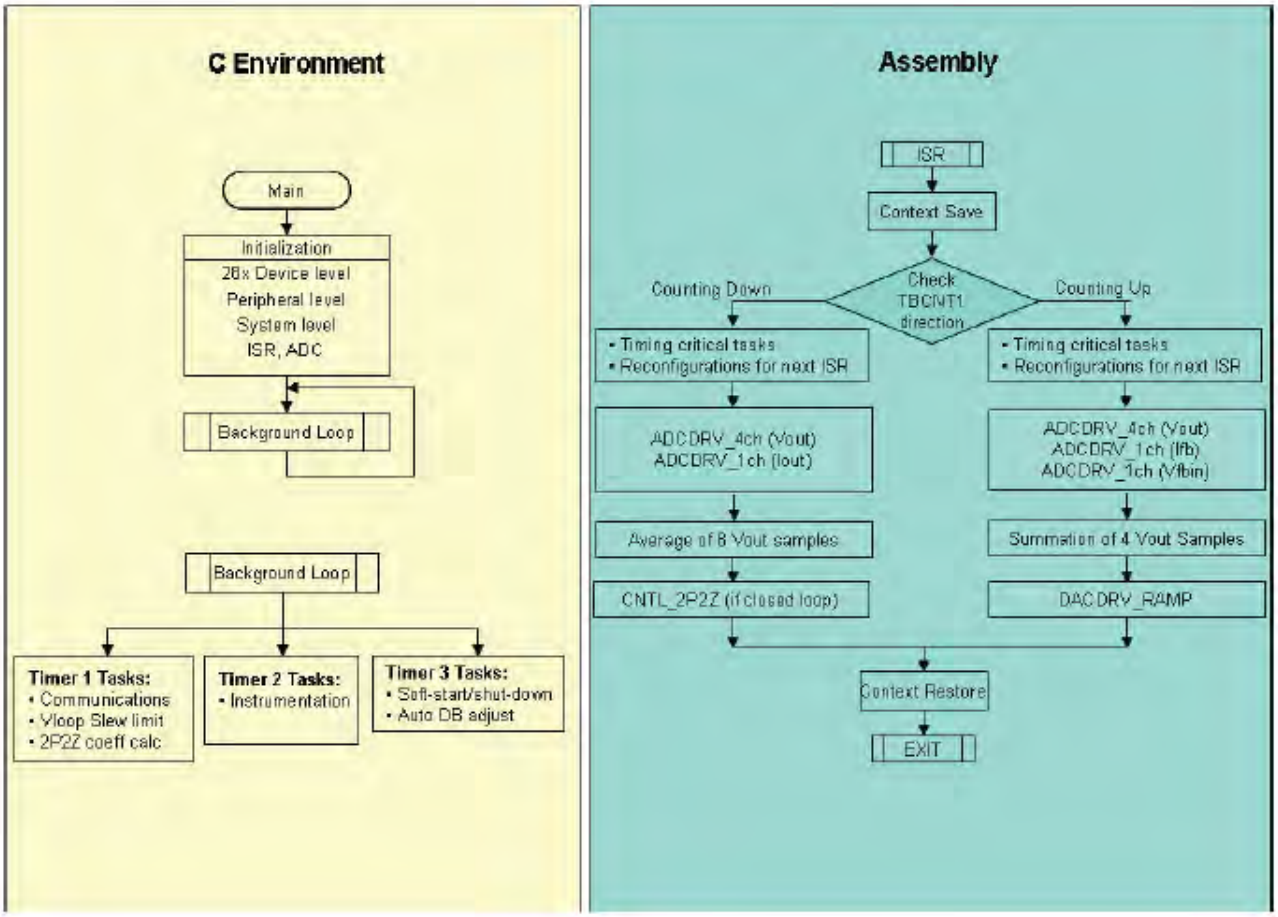


図 16 PCMCのソフトウェア・フロー

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ファイルは次の通りです。

HVPSFB-Main.c – アプリケーションの背後で「中枢部」となるファイルであり、アプリケーションの初期化、実行、管理に使用されます。

HVPSFB-DevInit.c – マイクロコントローラの一回限りの初期化と構成を担当します。このファイルには、クロック、PLL、GPIOのセットアップ等の関数などが入っています。

ISRは、次のファイルのみで構成されます。

HVPSFB-DPL-ISR.asm – このファイルは、すべての時間クリティカルな(all time critical)「制御タイプ」コードが入っています。このファイルには(一度だけ実行される)初期化セクションと、PWMスイッチング周波数の2倍で実行されるランタイム・セクションが入っています。

パワー・ライブラリ関数(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とアセンブリ両方のコンポーネントが含まれていることがあります。このプロジェクトでは、次に挙げるライブラリ・モジュールが使用されます。Cのモジュール名と、それに対応するアセンブリのモジュール名は次の通りです。

表 1 ライブラリ・モジュール

Cの構成関数	ASMの初期化マクロ	ASMのランタイム・マクロ
DAC_Cnf.c	DACDRV_RAMP_INIT n	DACDRV_RAMP n
ADC_SOC_Cnf.c	ADCDRV_4CH_INIT m,n,p,q	ADCDRV_4CH m,n,p,q
ADC_SOC_Cnf.c	ADCDRV_1CH_INIT n	ADCDRV_1CH n
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

制御ブロックは、次のように図解することも可能です (図17)。

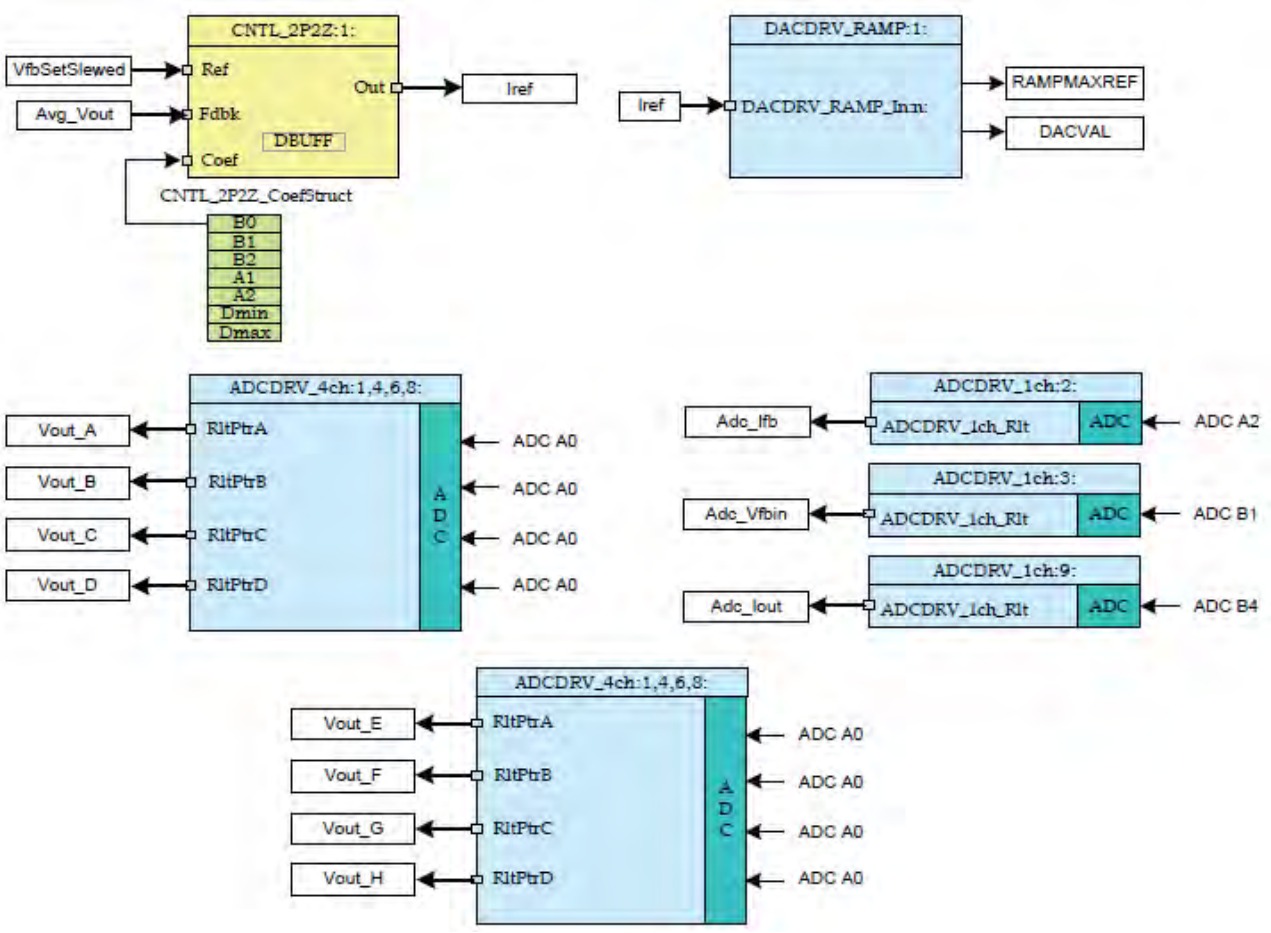


図17 PCMC ソフトウェアのブロック

図17のモジュールのカラー・コーディングに注目してください。「濃い目の青」のブロックはC2000マイクロコントローラのハードウェア・モジュールを表しています。「青」のブロックは、これらのモジュールのソフトウェア・ドライバです。「黄色」のブロックは制御ループのコントローラ・ブロックです。ここでは2-pole 2-zeroのコントローラを使用していますが、3-pole 3-zeroやPI/PID等、このアプリケーションに合うように実装できればどのコントローラでも問題ありません。図17のようなモジュール・ライブラリ構造体(modular library structure)を使用して、システム・ソフトウェア・フロー全体を視覚化し、理解しやすくしたものが図18になります。このような図により、各種機能を使用したり、追加/削除したりすることも容易になります。この事実は、インクリメンタル・ビルド・アプローチ(incremental build approach)を実装することにより、このプロジェクトで十分に証明されています。これは、次のセクションでさらに詳細に説明されます。

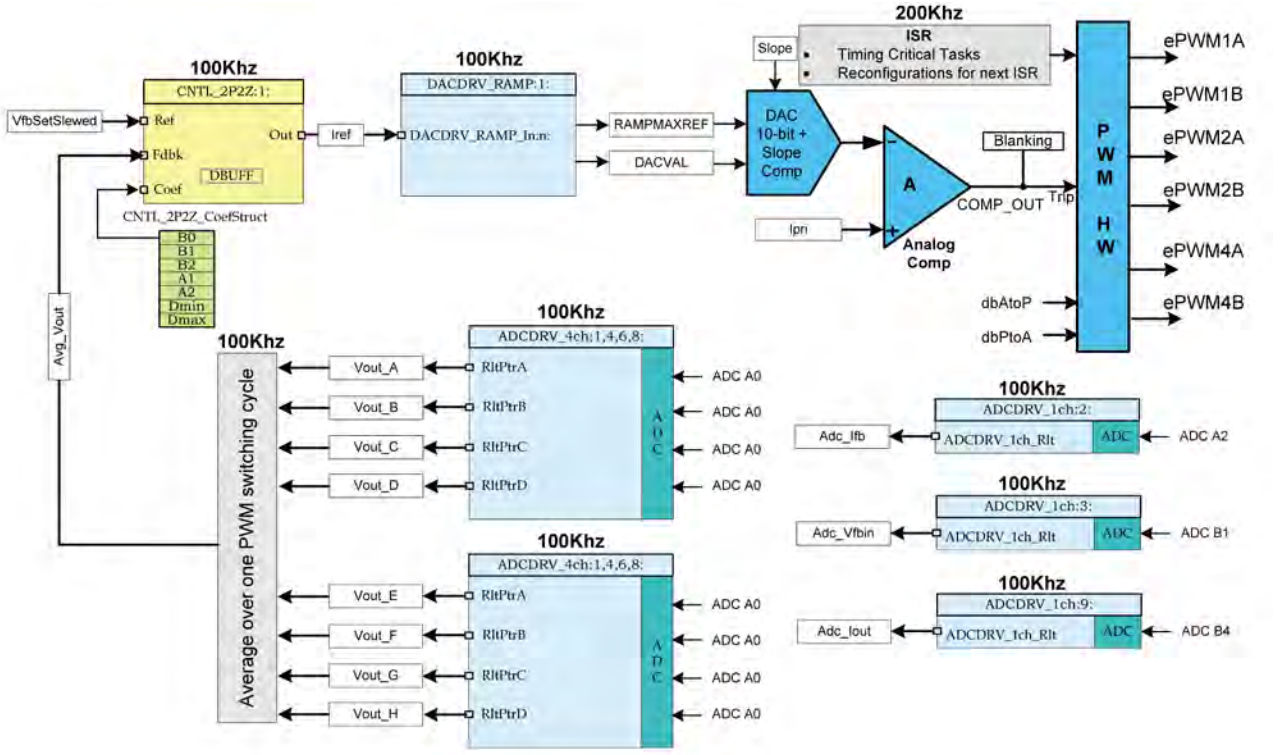


図 18 PCMC制御のフロー

システムは、2つのフィードバック・ループにより制御されます。一方は外側の電圧ループであり、ソフトウェア制御ブロックを使用して実装されます。もう一方は内側のピーク電流ループであり、オンチップのアナログ・コンパレータ、DAC、PWMの各種ハードウェア・リソースを使用して実装されます。図18には、各制御ブロックが実行される速度(rate)も示されています。例えば、電圧ループ・コントローラは100kHz(PWM周波数と同じ)という速度で実行されます。割り込みサービス・ルーチン(ISR)がPWM周波数の2倍で実行されることに注意してください。次に、上記で実装された制御について説明します。

出力電圧は、時間的に等間隔に分かれた4つのポイントで、つまり1/2PWMスイッチング・サイクルごとにサンプリングされます。これらの出力電圧サンプルの平均(Avg_Vout)は、1PWMサイクルについて計算されます。Avg_Voutが、電圧コントローラ内の基準電圧コマンド(Vref)をスルーレートで補正したもの(slewed version)(VfbSetSlewed)と比較されます。次に、電圧コントローラの出力が適切なDACコマンドに変換されます。これは、フルブリッジの2つのレッグ間の位相重複の量を検出して出力電圧を安定化させるピーク基準電流コマンドです。スロープ補償メカニズム用の「スロープ」値は、より低速のシステム・レベルのタスクにある制御フローの外側からプログラムすることが可能です。デフォルトのスロープ値として、最低値0.04V/usが提供されます([1]、[2]、[4]で行われる計算を参照してください)。オンチップのアナログ・コンパレータでは、トランスの1次電流を、スロープ補償されたピーク基準電流と比較します。コンパレータの出力により、デバイス上のPWMハードウェアが直接駆動されます。タイム・クリティカルな構成コードには、ISR内で実行されるものもあります。ISRは、1PWMサイクルにつき2回トリガされます。dbAtoPパラメータとdbPtoAパラメータでは、フルブリッジの「能動 – 受動(Active to Passive)」レッグと「受動– 能動(Passive to Active)」レッグにそれぞれデッド・バンド値を提供します。これらの値は、負荷範囲全体でZVS/LVSを実現するために使用されます。

3.2 インクリメンタル・ビルド

このプロジェクトは、2つのインクリメンタル・ビルドに分かれています。このことにより、ボードとソフトウェアについて学習し、慣れることが容易になっています。このアプローチはボードのデバッグ/テストにも有効です。

ビルドのオプション(選択肢)を次に示します。特定のビルド・オプションを選択するには、HVPSFB-Settings.h ファイルにあるマクロINCR_BUILDを、下の表に記載された対応するビルド選択項目に設定します。ビルド・オプションを選択した後、rebuild-all(すべてをリビルドする)のコンパイラ・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。次の章では、各ビルド・オプションの実行についてさらに詳細に説明します。

インクリメンタル・ビルドのオプション	
INCR_BUILD = 1	定数I(constant I)コマンドと開電圧ループを使用したピーク電流ループのチェック(PWM駆動回路と検知回路をチェック)
INCR_BUILD = 2	閉電流および閉電圧ループ(完全なPSFB)

表 2 PCMCのインクリメンタル・ビルド・オプション

4 インクリメンタル・ビルドを実行するための手順 - PCMC

PSFBシステムを作成するためのCフレームワーク用のメイン・ソース・ファイル、ISRアセンブリ・ファイル、プロジェクト・ファイルは、次に示すディレクトリにあります(最新のソフトウェア・パッケージを使用してください。2012年3月付のバージョン1.1が最新のソフトウェア・バージョンです)。

..\controlSUITE\development_kits\HVPSFB_v1.1\HVPSFB_PCMC

このソフトウェアに入っているプロジェクトは、CCSv4をターゲットとしています。

警告

ボード上は高電圧状態になっているため、ラボ環境の経験を積んだ電源の専門家以外はボードを取り扱わないようにしてください。このボードを安全に評価するには、絶縁処理を施した適切な高電圧DC電源を使用する必要があります。ボードにDC電力が印加される前に、電圧計と適切な抵抗負荷または電気的負荷(resistive or electronic load)を出力に取り付けてください。電力が印加されている間は、絶対にユニットに触れないでください。

次に示すステップに従って、HVPSFB_PCMCソフトウェアに含まれている例をビルドして実行してください。

4.1 ビルド 1: 開電圧ループを使用したピーク電流ループのチェック

➢ 目的

このビルドの目的は、システムのピーク電流モードの動作を評価し、DACおよびADCドライバ・モジュールを検証し、ボード上のMOSFETドライバ回路と検知回路を検証し、Code Composer Studio (CCS)の動作に慣れることです。このシステムは開ループで動作するため、ADCで計測された値はこのビルドでは計測(instrumentation)の目的でのみ使用されます。プロジェクトのビルドと実行に必要なステップを以降で紹介していきます。

➢ 概要

ビルド1のソフトウェアは、オシロスコープで様々な波形を見て、このコマンドをCCSからインタラクティブに補正することで発生する出力電圧のピーク基準電流コマンドの変化の影響を観察することにより、ユーザーがDACドライバ・モジュールを短時間で評価できるように構成されています。また、ADCでサンプリングされたデータをウォッチ・ビューで見ることで、ユーザー側でADCドライバ・モジュールを評価することが可能です。

前章で述べたように、DACドライバとADCドライバのマクロのインスタンス化は_DPL_ISR内部で実行されます。図19は、このビルドで使用するブロックです。ピーク基準電流コマンドは、DACDRV_RAMPモジュールに書き込まれます。このモジュールでは、適切な16ビット値(RAMPMAXREF))を駆動します。この値は、スロープ補償に使用されるRAMP(ランプ波)の開始値です。このモジュールではまた、DACVALレジスタに対して適切な10ビット値を導出します。この値は、スロープ補償が必要でないか、外部的に提供される場合に使用できます。オンチップのアナログ・コンパレータでは、トランスの1次電流を、スロープ補償されたピーク基準電流と比較します。コンパレータの出力は、PWMモジュールのトリップ・ゾーン・ロジックに接続されます。ePWM1モジュールは、システムのマスタ・タイムベースとして機能し、アップダウン・カウント・モードで動作しますが、他のPWMモジュールはアップカウント・モードで動作します。ePWM1AおよびePWM1BではQ1とQ4のフルブリッジ・スイッチを駆動しますが、ePWM2AとePWM2BではQ2とQ3のフルブリッジ・スイッチを駆動します。ePWM4AとePWM4BではQ5とQ6の同期整流器スイッチを駆動します。1回のPWMハーフ・サイクル(PWM half cycle)内でコンパレータの出力がHighになると常に、その瞬間にHighだったePWM2モジュールの出力(ePWM2AまたはePWM2B)が即座にLowにプルされますが、もう一方のPWM2モジュールの出力は適切なデッドバンド・ウィンドウ(dbAtoP)の後でHighにプルされます。ePWM4Aの出力とePWM4Bの出力も同様に駆動されます。これらの波形を図12に示します。

注意する必要があるのは、図19の濃い青のブロックに示されているように、このスロープ補償のランプ波の生成、コンパレータの動作、PWM波形の生成はすべて、ソフトウェアの関与なしにハードウェア的に行われるということです。次の1/2PWMサイクルに備えて、ISR内部である程度のレジスタ再構成が行われます。この時間的にクリティカルな(time critical)コードはISRの開始時に実行され、再度命令することや変更することはできません。アセンブリのISR _DPL_ISRルーチンは、ePWM1によりトリガされます。ISRのトリガ周波数は、PWMのスイッチング周波数のトリガ周波数の2倍であることに注意してください。

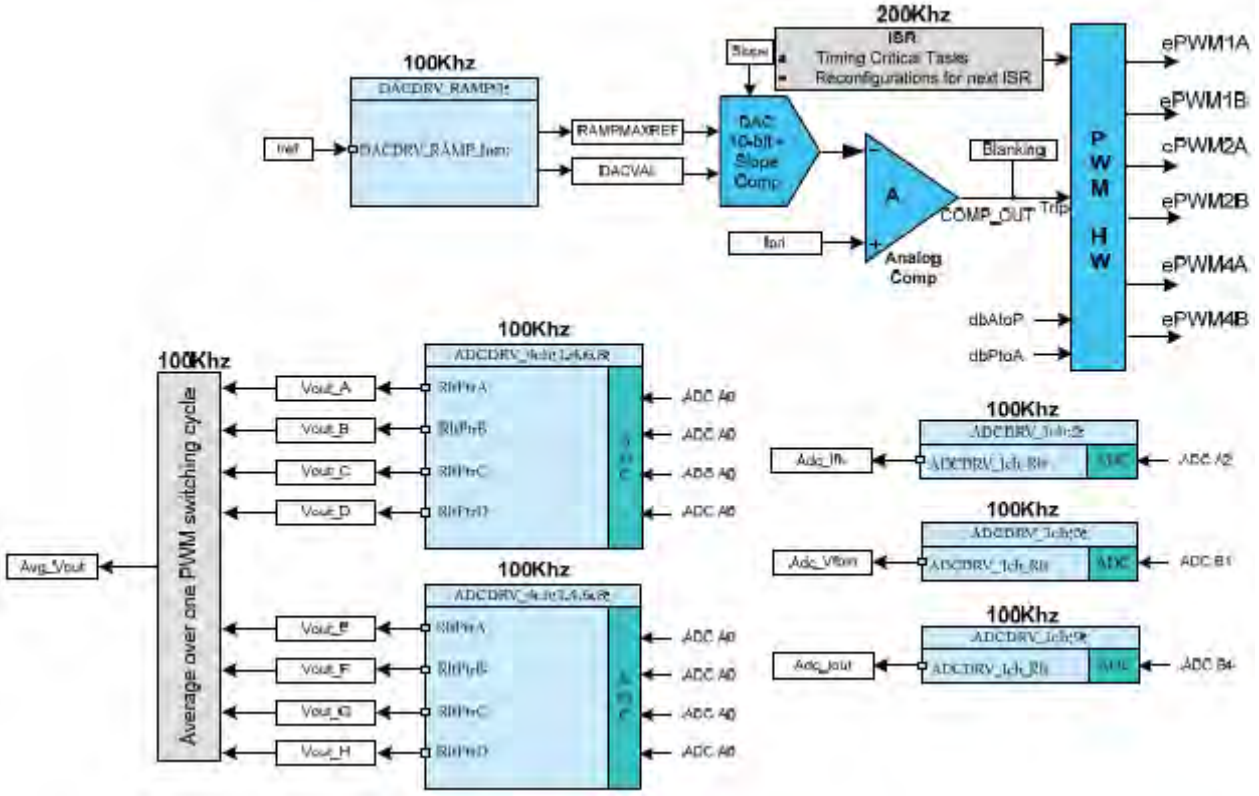


図19 ビルド1のソフトウェア・ブロック

RAMPMAXREF値とDACVAL値は、入力「Iref」(Q24の変数)コマンドから導出(derive)されます。表3には、「Iref'」から計算されたRAMPMAXREF値とDACVAL値の例が記載されています。

表 3 RAMPMAXREF値とDACVAL値の例

Iref (Q24)	RAMPMAXREF = (Iref/2^8)	DACVAL = (Iref /2^14)	ピーク基準電流 (最大値 = 6.776A)
2097152d	8192	128	0.845A
8388608d	32768	512	3.39A
16776704d	65534	1023	6.77A

ADCドライバ・モジュールは、12ビットのADC結果を読み取って、Q24の値に変換するために使用されます。1/2PWMサイクルごとに、PWM1 SOCA (変換Aの開始)、PWM3 SOCA (変換Aの開始)、PWM3 SOCB (変換Bの開始)を使用して、出力電圧用に4回分のADC変換がトリガされます。これらの変換のトリガ・ポイントは、時間的に等間隔に分かれています。この4回分の結果はISRごとに読み出され、各PWMサイクル内で出力電圧のADC変換結果を合わせて8回分提供します。ISR中のコード数行を使用して、これらの8つの結果に基づき、PWMサイクル1回分の平均出力電圧を計算します。

出力電圧検知回路は、シンプルな分圧回路で構成されています。巻き数比1:100の電流トランス、センス抵抗(sense resistor)(48.7オーム)を使用して、フルブリッジ・トランスの1次電流を検知します。入力電圧の検知は、補助パワー・モジュール(J7)により実現されます。これらの計算の詳細については、HVPSFB-Calculations.xlsファイルを参照してください。

保護機能(Protection)

この段階で、ボードで使用されているシャットダウン機構を紹介することが適切と思われます。ボードには、オンチップのアナログ・コンパレータ2を使用した、トランスの1次電流用の過電流保護機能が実装されています。基準トリップ・レベルは、内蔵の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を構成して、検知された電流が設定された限界値より大きい場合は常に、ePWM1上で1回限りの(ワンショット)トリップ・アクションを生成するようにします。C2000デバイス上のトリップ機構の柔軟性により、様々なトリップ・イベント時に様々なアクションを実行することが可能です。このプロジェクトでは、ePWM1Aの出力とePWM1Bの出力が即座にLowに駆動されることで、電力段(power stage)が保護されます。ePWM2A、ePWM2B、ePWM4A、ePWm4Bの各出力がその後、ソフトウェアから強制的にLowに駆動されます。デバイスのリセットが実行されるまで、すべての出力がこの状態(ステート)に保持されます。これらの計算の詳細については、HVPSFB-Calculations.xlsファイルを参照してください。

定電流モードまたは定電力モードで動作している場合には、出力低電圧シャットダウン機構もソフトウェアから実装されます。

入力低電圧および入力過電圧ロックアウトも、ソフトウェア的に実装されます。

リソースのマッピング(割り当て)

C2000のMCU～HVPSFB段間の主要な信号接続を、次の表にまとめてあります。PWMのマッピングがVMCプロジェクトとPCMCプロジェクトでは異なることに注意してください。コントローラ・カード上のジャンパ(J2、J3 – PCMCおよびVMCのPWM駆動ジャンパのイネーブル)を、VMCモード用に正しく構成する必要があります(デフォルトのジャンパ位置は、PCMC動作用にセットされています)。2つのモード用のジャンパ構成を次に紹介します。

- **PCMC**: J2(1) -> J3(1)、J2(2) -> J3(2) 、J2(3) -> J3(3) 、J2(4) -> J3(4) 、J2(7) -> J3(7) 、J2(8) -> J3(8) 、
- **VMC**: J2(1) -> J3(3) 、J2(2) -> J3(4) 、 J2(3) -> J3(1) 、 J2(4) -> J3(2) 、 J2(7) -> J3(8) 、 J2(8) -> J3(7)

表 4 HVPSFB信号インターフェイスの参考表(リファレンス) – PCMC

信号名	説明	C2000コントローラへの接続
ePWM-1A	PWM駆動(フルブリッジ・スイッチQ1用)	GPIO-00
ePWM-1B	PWM駆動(フルブリッジ・スイッチQ4用)	GPIO-01
ePWM-2A	PWM駆動(フルブリッジ・スイッチQ2用)	GPIO-02
ePWM-2B	PWM駆動(フルブリッジ・スイッチQ3用)	GPIO-03
ePWM-4A	PWM駆動(同期整流器スイッチ Q5用)	GPIO-06
ePWM-4B	PWM駆動(同期整流器スイッチ Q6用)	GPIO-07
Vout	PSFB出力電圧	ADC-A0
Ifb	トランスの1次電流	ADC-A2/COMP1A
Ifb	トランスの1次電流	ADC-A4/COMP2A*
Vfbin	PSFBの入力電圧	ADC-B1
Iout1	PSFBの出力電流	ADC-B3
Iout2	PSFBの出力電流(大幅にフィルタ処理)	ADC-B4

* コントローラ・カード上のデフォルトのジャンパJ4構成です。入力はジャンパJ4を選択することで構成可能です。

注: HVPSFB_PCMCプロジェクトとHVPSFB_VMCプロジェクトでは、device_supportディレクトリに配置されているファイルではなく、それら自体のプロジェクト・ディレクトリに配置されているDSP2802x_Comp.hファイルとDSP2802x_EPWM.h ヘッダ・ファイルを使用します。device_support ディレクトリ内の2つのファイルは、後でControlSuiteがアップデートされる際にアップデートされます。アップデート時には、これらのアップデートされたファイルを2つのプロジェクトに使用できます。

➢ 手順(Procedure)

CCSを起動してプロジェクトを開く

このビルドを短時間で実行するには、次のステップに従います。

1. ベース・ボード上でジャンパJ8とJ6が装着されていることを確認します。
2. デフォルトでは、コントローラ・カードのPiccolo Macro上の抵抗R6とジャンパJ1を外して、FLASHからの起動をイネーブルにします。RAMの実行とプログラミング、またはFLASHのプログラミングを行うには、この2つを再度装着(Re-populate)します。

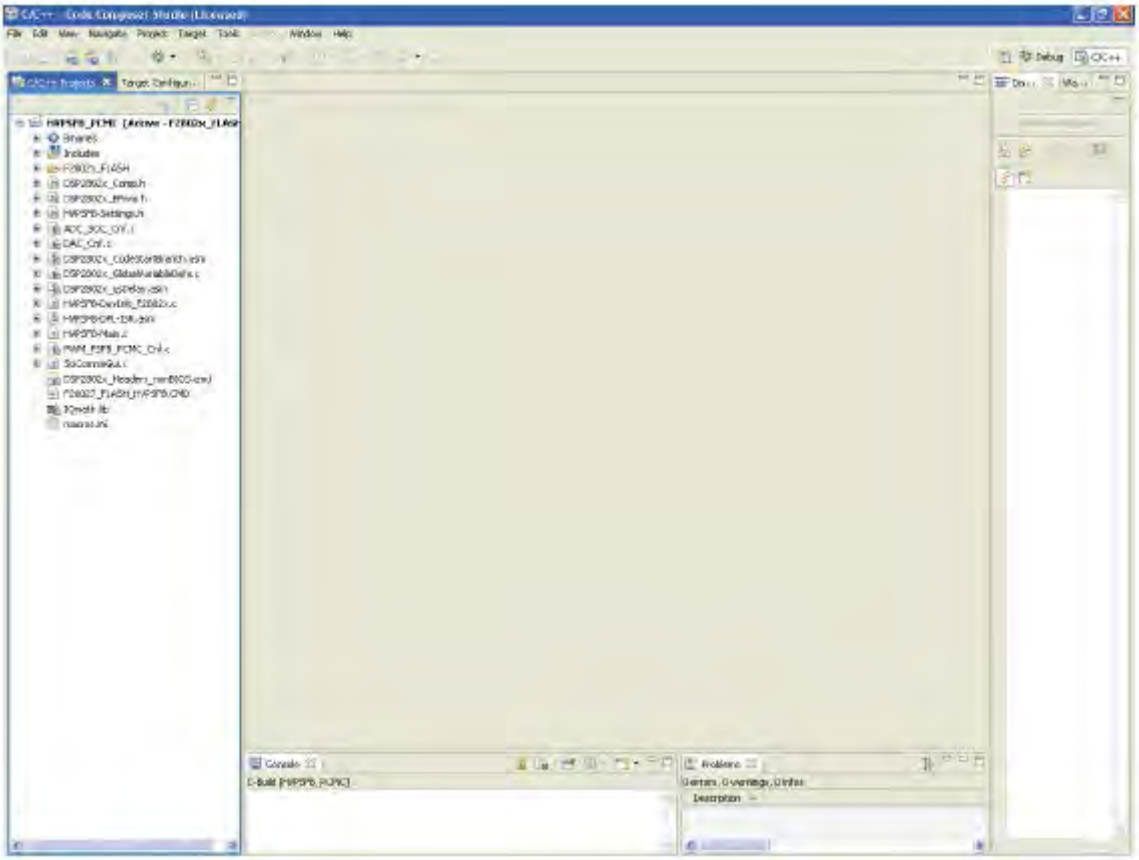
USBコネクタを、エミュレーション用のPiccoloコントローラ・カードに接続します。DC約400Vを出力するためにセットされた、適切な絶縁処理済みのDC電源のご使用が推奨されます。DC電源がメイン・ボード上のJ1とJ2に接続されている場合は、DC電源をオフにしておく必要があります。20AWGの配線600Vを使用して、電源をJ1とJ2に接続します。この接続の極性が正しいことを確認してください。適切な抵抗負荷/DC電気的負荷を、位相シフト・フルブリッジ・システムのDC出力側のJ3とJ4に取り付けます。この時点ではDC電源400Vをオンにしないでください。TP1～TP2間のバイアス電源を、DC約11V(この電圧は12Vより低くする必要があります)を使用して起動します。

3. デスクトップのCode Composer Studioアイコンをダブルクリックします。Code Composer Studioを最大表示にして、お使いのディスプレイ全面に表示します。Welcome 画面が開いた場合は閉じてください。

4. プロジェクトには、MCUハードウェア上で動作させることのできる実行可能な出力ファイル(.out)の開発に必要なすべてのファイルとビルド・オプションが入っています。メニュー・バー上で、「Project」 -> 「Import Existing CCS/CCE Eclipse Project (既存のCCS/CCE Eclipseプロジェクトをインポートする)」をクリックして、「Select root directory (ルート・ディレクトリの選択)」から「..\controlSUITE\development_kits\HVPSFB_v1.1\HVPSFB_PCMC」ディレクトリを探して選択します。「Projects」タブ下で、HVPSFB_PCMCにチェックが入っていることを確認してください。「Finish」をクリックします。

   このプロジェクトでは、プロジェクトをビルドするために必要なすべてのツール(コンパイラ、アセンブラ、リンカ)を呼び出します。

5. 左側のプロジェクト・ウィンドウで、プロジェクトの左側にあるプラス記号(+)をクリックします。お使いのプロジェクト・ウィンドウが次のように表示されます。



Device Initialization (デバイスの初期化)ファイル、Main(メイン)ファイル、ISRファイル

注: ソース・ファイルは絶対に変更せず、内容を見て調べるだけにしてください。

6. プロジェクト・ウィンドウのファイル名をダブルクリックして、HVPSFB-DevInit_F2802x.c を開いて調べてください。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、ペリフェラル・クロックのイネーブルがセットアップされていることを確認した後、共有のGPIOピンが構成されていることを確認してください。

7. HVPSFB-Main.cを開いて調べてください。DeviceInit()関数に対して行われた呼び出しと、その他の変数初期化を確認してください。また、各種インクリメンタル・ビルド・オプション(具体的には現時点でコンパイルしようとしているビルド)用のコード、ISR初期化、(;;) ループ用のバックグラウンドも確認してください。

8. メイン・ファイル内で、ビルド1の固有の初期化コードの下にある、次に示すコードを探して調べてください。このコードでは、制御フロー内でDACDRV_RAMPブロックが接続され、初期化されます。

```
DacDrvCnf(1, 1280, 1, 2, Slope);        // Comp1, DACval = 1280(Initial), Slope compensation is used,
                                        // Ramp is PWM3 Synced, Initial Slope
// DAC connections
    DACDRV_RAMP_In1 = &Iref;            // Controls the DAC reference voltage
```

9. メイン・ファイル内で、ビルド1の固有の初期化コードの下にある、次に示すコードを探して調べてください。このコードでは、制御フロー内でADCDRV_4CHブロックと、ADCDRV_1CHブロックの複数のインスタンス化(instantiation)が構成され、初期化され、 接続されます。

```
#define     Vfb_outR        AdcResult.ADCRESULT1    //
#define     IfbR            AdcResult.ADCRESULT2    //
#define     Vfb_inR         AdcResult.ADCRESULT3    //

#define     IoutR           AdcResult.ADCRESULT9    //

// Channel Selection for Cascaded Sequencer
    ChSel[0] = 0;               // A0 - O/P Voltage - Dummy
    ChSel[1] = 0;   //B        // A0 - O/P Voltage
    ChSel[2] = 2;               // A2 -  Transformer Primary Current
    ChSel[3] = 9;               // B1 -  I/P Voltage
    ChSel[4] = 0;   //C        // A0 -  O/P Voltage

    ChSel[5] = 0;               // A0 -  O/P Voltage - Dummy
    ChSel[6] = 0;   //A        // A0 -  O/P Voltage
    ChSel[7] = 0;               // A0 -  O/P Voltage - Dummy
    ChSel[8] = 0;   //D        // A0 -  O/P Voltage
//  ChSel[9] = 11;              // B3 - Iout1
    ChSel[9] = 18;              // B4 - Iout2

    TrigSel[0] = ADCTRIG_EPWM3_SOCA;        // O/P Voltage sampling triggered by EPWM3 SOCA - Dummy
    TrigSel[1] = ADCTRIG_EPWM3_SOCA; //B    // O/P Voltage sampling triggered by EPWM3 SOCA
    TrigSel[2] = ADCTRIG_EPWM3_SOCA;        // Transformer Primary Current sampling triggered by EPWM3 SOCA
    TrigSel[3] = ADCTRIG_EPWM3_SOCA;        // I/P Voltage sampling triggered by EPWM3 SOCA
    TrigSel[4] = ADCTRIG_EPWM3_SOCA; //C    // O/P Voltage sampling triggered by EPWM3 SOCA

    TrigSel[5] = ADCTRIG_EPWM1_SOCA;        // O/P Voltage sampling triggered by EPWM1 SOCA at CTR = ZRO or PRD - Dummy
    TrigSel[6] = ADCTRIG_EPWM1_SOCA; //A    // O/P Voltage sampling triggered by EPWM1 SOCA at CTR = ZRO or PRD
    TrigSel[7] = ADCTRIG_EPWM3_SOCB;        // O/P Voltage sampling triggered by EPWM3 SOCB at CMPB3 - Dummy
    TrigSel[8] = ADCTRIG_EPWM3_SOCB; //D    // O/P Voltage sampling triggered by EPWM3 SOCB at CMPB3

    TrigSel[9] = ADCTRIG_EPWM2_SOCA;        // Iout triggered by EPWM2 SOCA

    EALLOW;
    AdcRegs.SOCPRICTL.bit.SOCPRIORITY = 9;  // SOC0-8 are high priority
    EDIS;
    ADC_SOC_CNF(ChSel,TrigSel,ACQPS, 16, 0);// ACQPS=8, No ADC channel triggers an interrupt IntChSel > 15,
                                            // Mode= Start/Stop (0)
```

10. HVPSFB-DPL-ISR.asmを開いて調べてください。_DPL_Initと _DPL_ISRのセクションを確認してください。ここでは、DACドライバとADCドライバのマクロのインスタンス化が、初期化用とランタイム用にそれぞれ行われます。オプションとして、調べたファイルを 閉じることもできます。

プロジェクトのビルドとロード(Build and Load the Project)

11. HVPSFB-Settings.h ファイルの1、つまりインクリメンタル・ビルド・オプションを選択します。

注: HVPSFB-Settings.hのインクリメンタル・ビルド・オプションを変更した場合は必ず、「Rebuild All(すべてをリビルドする)」を行ってください。

12. Project ->「Rebuild All(すべてをリビルドする)」ボタンをクリックして、ツールがビルド・ウィンドウで実行されるのを確認します。

13. Target ->「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。まだターゲット構成ファイルが選択されていない場合は、新規ターゲット構成ファイルを開くように指示するCCSのメッセージが表示されます。有効なターゲット構成ファイルがこの接続用に生成済みの場合は、ステップ15に進んでください。新規ターゲット構成ウィンドウで「.ccxml」という拡張子の付いたファイル名(例: xds100-F28027.ccxml)を入力して、今回の作業を行うターゲットの名前とします。「Use shared location (共有の場所を使用)」をチェックして、Finish(終了)をクリックします。

14. .ccxmlファイルを開いて、Connection(接続)として「Texas Instruments XDS100v2 USB Emulator」をクリックし、そのデバイス名の下でスクロールダウンして「TMS320F28027」を選択します。「 Save(保存)」をクリックします。

15. Target -> 「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。選択されたプロジェクト構成に応じて、プログラムがFLASHメモリまたはRAMメモリにロードされます。このプロジェクトは、F2802x_FLASHの構成でのみ提供されます。これで、Main() を開始する準備ができました。

デバッグ環境のウィンドウ(Debug Environment Windows)

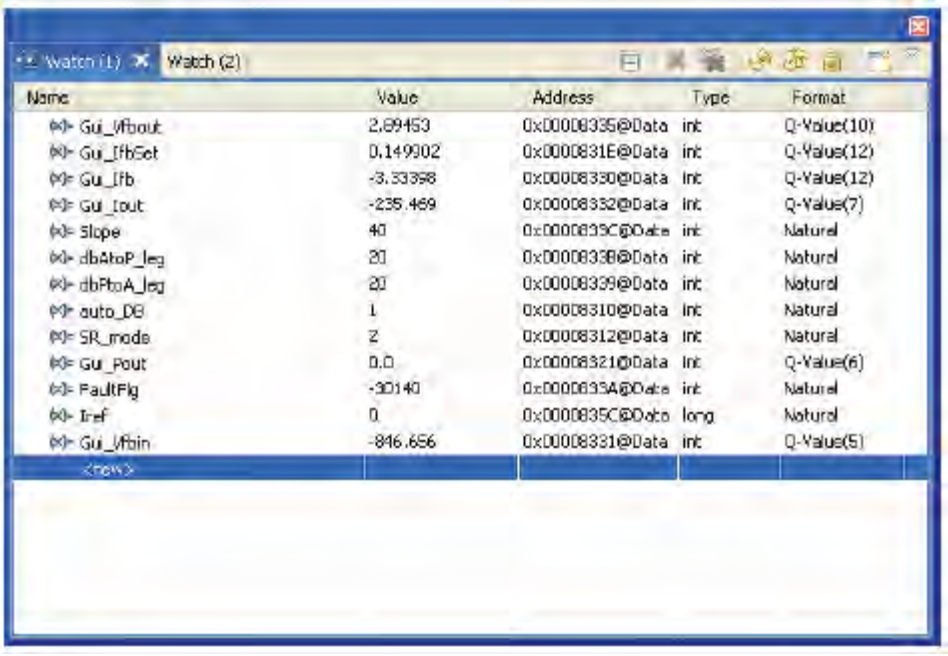
コードのデバッグ中には、ローカル変数とグローバル変数を監視するのが一般的です。Code Composer Studioでこの監視を行うには、メモリ・ビューやウォッチ・ビューなど様々な手段があります。さらに、Code Composer Studioには時間領域(および周波数領域)のプロットの作成機能があります。これにより、グラフ・ウィンドウを使用して波形を見ることも可能です。

16. デバッグ環境が起動してもウォッチ・ビューが開かない場合は、新規のウォッチ・ビューを開き、次の手順に従って各種パラメータを追加してください。

    メニュー・バーで、View(表示) -> Watchをクリックします。

    「Watch (1)」タブをクリックします。ウォッチ・ビューに任意の変数を追加できます。「Name」カラムの空白のボックスに監視対象の変数のシンボル名を入力し、キーボードからエンター・キーを押します。必要に応じて、「Format」を修正してください。ウォッチ・ビューは次のような外観になっています。この時点では初期化されていない変数もメイン・コード内に存在するため、変数に不要なデータが含まれている可能性があることに注意してください。

    (設定されている場合)FaultFlgは(上記で説明された)過電流条件を示しており、この条件によりPWM出力がシャットダウンされます。 デバイスのリセット(ステップ32に従って正しく行うこと)まで、PWM出力はこのステートに保持されます。Ipri_trip変数により、オンチップ・コンパレータ2用に内部10ビットDAC基準レベルがセットされます。これがQ15の数であることに注意してください。



リアルタイム・エミュレーションを使用する

リアルタイム・エミュレーションは、*MCUの実行中に*Code Composer Studio内のウィンドウのリフレッシュ・レートを最大10Hzにアップデートすることを可能にする、特別なエミュレーション機能です。これにより、グラフやウォッチ・ビューがアップデート可能になるだけでなく、ユーザーがウォッチ・ウィンドウやメモリ・ウィンドウの値を変更して、MCUの動作にその変更を反映できるようになります。これは、制御のロウ・パラメータ(raw parameters)をオンザフライで調整する場合などに非常に役立ちます。

17. リアルタイム・モードをイネーブルにするには、マウスポインタを水平ツールバー上のボタンの上に置き、次のボタンをクリックします。  Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)
18. メッセージ・ボックスが表示されることがあります。その場合は「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」にセットされます。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」にセットされると、メモリ値とレジスタ値がホスト・プロセッサに渡されて、デバッガのウィンドウがアップデートできる ようになります。
19. ここで、同じ水平ツールバー上の次のボタンをクリックします。

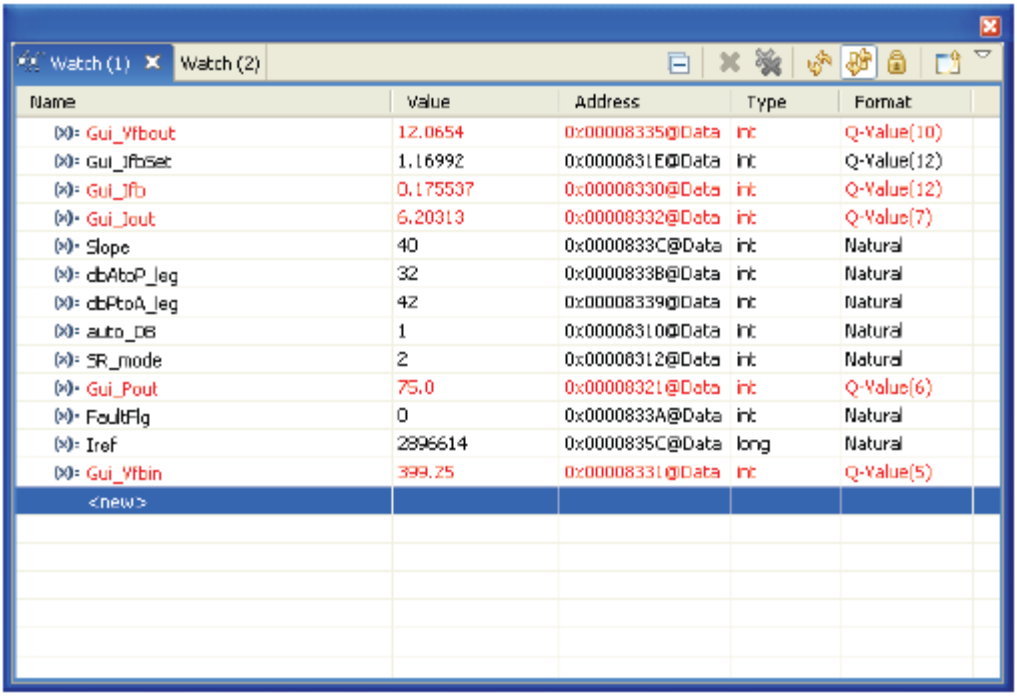

20. 多数のウィンドウが開いた場合、エミュレーション・リンクに関連する帯域幅に制限があるため、継続的なリフレッシュでアップデートされるウィンドウと変数が多くなりすぎて、リフレッシュ周波数が完全に停止する可能性があります。ウォッチ・ビューの  ボタンを右クリックして、「Customize Continuous Refresh Interval..(継続的なリフレッシュ間隔をカスタマイズする)」を選択します。「Continuous refresh interval(milliseconds) (継続的なリフレッシュ間隔(ミリ秒単位))」の値を変更して、ウォッチ・ビューの変数の リフレッシュ・レートを下げます。通常では、4000msのリフレッシュ・レートであれば、このアプリケーション・ノートの演習用としては 十分です。
21. ウォッチ・ビュー用に、「 Continuous Refresh(継続的なリフレッシュ)」ボタン  をクリックします。

コードの実行

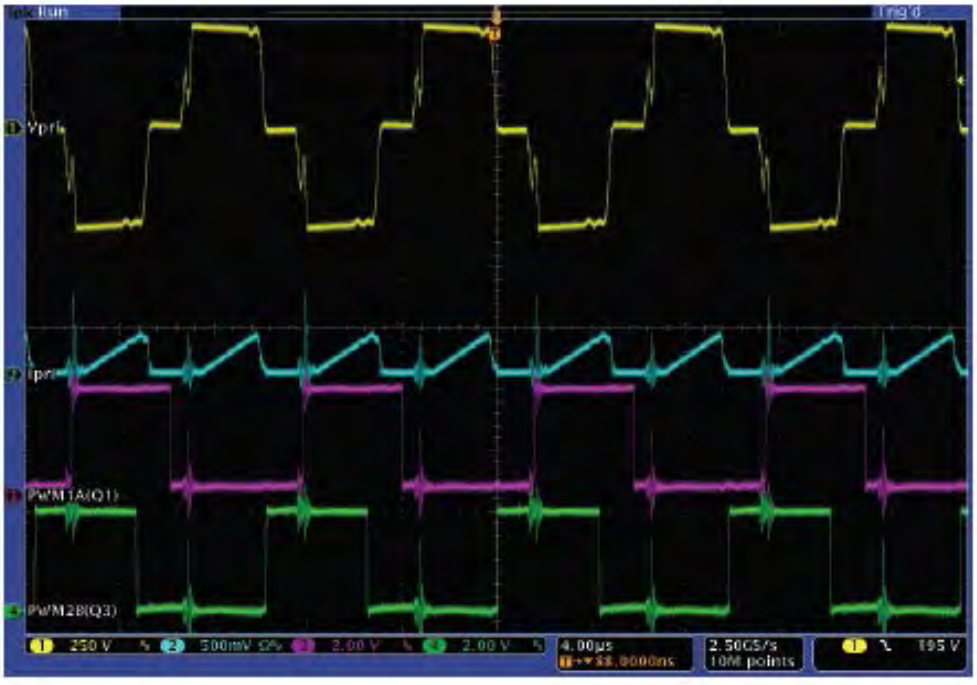
22. <F8>キー、ツールバーのRunボタン、メニュー・バーのTarget-> Runのどれかを使用して、コードを実行します。
23. ウォッチ・ビューで、変数Gui_IfbSetを0.15 (Q12)に設定する必要があります。この変数は、ピーク基準電流コマンドをアンペア単位で示し、IrefをDACDRV_RAMPモジュールに対して駆動します。Gui_IfbSetには、0.15より小さい値を使用しないでください。
24. 適切な抵抗負荷をPSFBシステムのDC出力側に取り付けます。出力12Vで約3A～6Aの電流を引き出す負荷は、良い開始点となります。

注: 安全上の理由から、絶縁DC電源を使用してDC入力400Vをボードに供給することが推奨されます。

25. J1、J2の入力に、400VのDCで電力を供給します。
26. ウォッチ・ビューでGui_IfbSetを前より高い値(例: 0.25)に設定して、ピーク基準電流コマンドを大きくすると、出力電圧が増大します。出力電圧を注意して観察し、ボードの許容範囲を超えないようにしてください。ある一定のGui_IfbSet値を使用して動作している場合は、負荷が急に小さくなると出力電圧が上昇することを覚えていてください。したがって、ビルド1での動作中には、負荷を急激に変化させたり、Gui_IfbSetコマンドの数値を大幅に上げたりしないでください。
27. ウォッチ・ビューの様々なADC結果の、様々なGui_IfbSet値を観察します。
28. 入力電圧が約400Vで、出力12Vでの負荷が約6Aの、1.17Aという値のGui_IfbSeコマンドを使用するシステムの動作に対応するウォッチ・ビューを次に示します。



29. 次に示すオシロスコープのキャプチャは、上記で説明された条件下で見られるトランスの1次電圧、検知された1次電流(primary sensed current)、2つの対角線上で向き合ったスイッチ(Q1とQ3)を駆動するPWM波形を示しています。



30. デフォルトでは、同期整流器はモード2で駆動されます。同期整流器の動作のモードは、ウォッチ・ビューからSR_mode変数を0、1、2に変えることで変更できます。各種SRモードでの、引き込まれる入力電流の量の変化と出力電圧の変化を観察してください。同期整流器のスイッチを駆動するPWM波形を精査することもできます。非常に低い負荷で動作しているか、出力電圧が非常に低い(6Vを下回る)場合は、各種SRモードを変更しないでください。このような例では、デフォルトのSRモード2を使用してください。

31. 様々なGui_IfbSet値を試して、対応するADC結果を観察したり、少しずつGui_IfbSetを増加させたりしてみます。出力電圧を常に注意して観察し、ボードの許容範囲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

圧と入力電流、出力電圧等の様々な波形を精査することも可能です。この絶縁DC-DCコンバータについて、上記のような高電圧大電流を精査する際には、適切な安全対策を取り、適切なグラウンド要件について考慮する必要があります。

32. リアルタイム・モードのMCUを完全に停止するには、2ステップのプロセスを実行します。DC入力400Vをオフにして数秒待機します。最初に、ツールバーのHaltボタンか Target -> Halt を使用してプロセッサを停止します。次に、  ボタンを再度クリックしてMCUをリアルタイム・モードから解除してから、MCUをリセットします。

33. Code Composer Studioは、次の演習用に動作させておくことも、任意で閉じることもできます。

演習の終了

4.2 ビルド 2: 完全な HVPSFB

➢ 目的

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

➢ 概要

図20は、このビルドで使用するソフトウェア・ブロックです。電圧ループには2-pole 2-zeroのコントローラが使用されています。アプリケーションの制御ループの要件によっては、他のコントローラ・ブロック(3-pole 3-zeroやPI等)の使用も可能です。図に示すように、電圧ループ・ブロックは100KHzで実行されます。CNTL2P2Zは、IIRフィルタ構造体により実現される2次補償回路(2nd order compensator)です。この関数(機能)はどのペリフェラルからも独立しているため、CNF関数の呼び出しは必要としません。

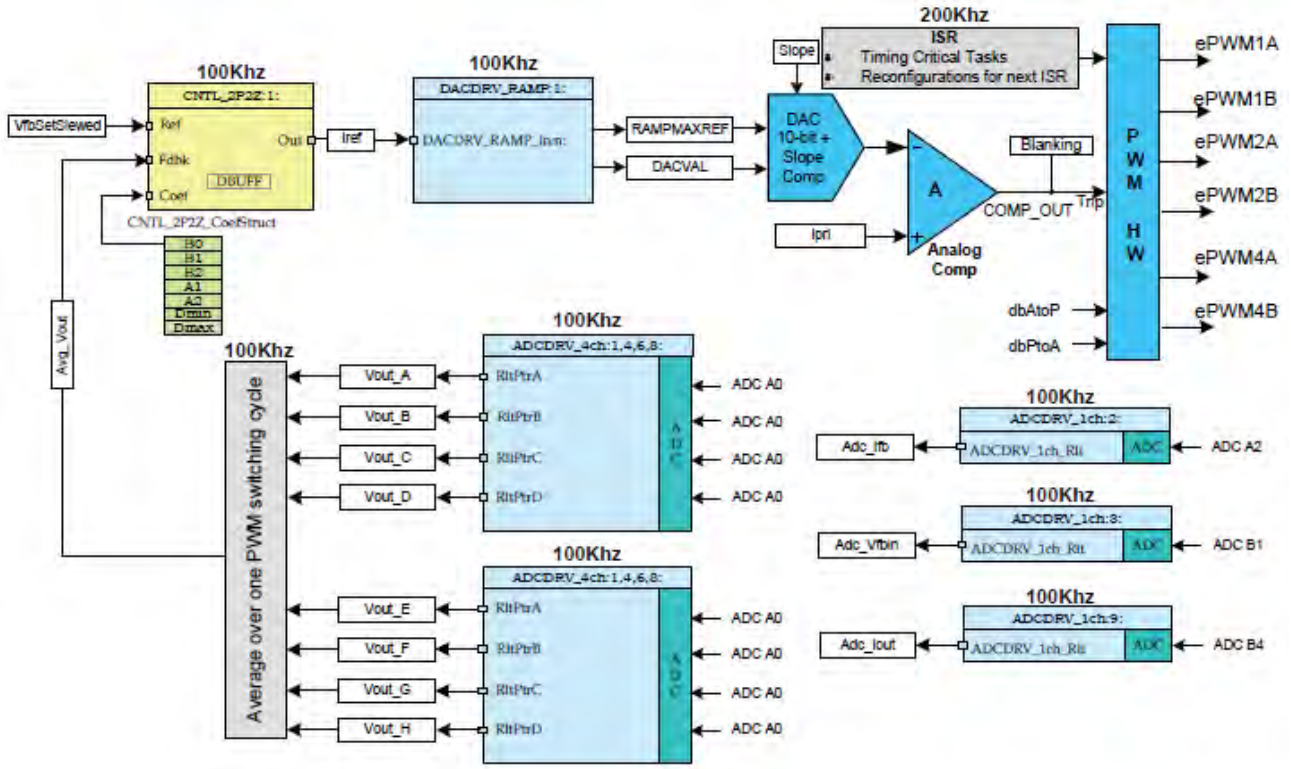


図20 ビルド2のソフトウェア・ブロック

修正する必要のある5つの係数(coefficients)が、構造体CNTL_2P2Z_CoefStruct1のエレメントとして格納されています。この構造体の他のエレメントは、コントローラ出力をクランプ(制限)するために使用されます。システムで複数のループが必要な場合は、CNTL_2P2Zブロックを複数回インスタンス化(instantiate)できます。各インスタンスでは別個の係数セットを持つことが可能です。5つの係数を試行錯誤しながら個別に直接扱うことはほぼ不可能であるため、数学的な解析(数理解析)および/またはmatlabやmathcad等のツールによる支援が必要になります。これらのツールではボード線図、根軌跡(root-locus)等の機能を使用して、位相マージン、ゲイン・マージン等を決定できます。

ループ・チューニングをシンプルに保ち、複雑な数学ツールや解析ツールを使用せずにすむようにしておくために、より直観的に理解しやすいP、I、Dという係数ゲイン(coefficient gains)をB0、B1、B2、A1、A2に効率的にマッピングすることにより、係数選択の問題の自由度のレベルが5段階から3段階に縮小されました。これにより、P、I、Dが別個かつ段階的に補正できるようになっています。このマッピングの式を次に示します。

補償回路ブロック(CNTL_2P2Z)には2つの極と2つの零点があり、汎用IIRフィルタの構造体がベースとなっています。また、基準入力とフィードバック入力もあります。電圧ループの場合は、1PWMサイクル(Avg_Vout)について計算されたf平均出力電圧がフィードバックになりますが、コントローラに対する基準入力は出力基準電圧コマンド(Vref)をスルーレートで補正したもの(slewed version) (VfbSetSlewed) になります。伝達関数は次の式で求められます。

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

PIDコントローラ の再帰形式(recursive form)は、次の差分方程式(difference equation)により求められます。

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

ここで、次のようになります。

$$b_0 = K_p' + K_i' + K_d'$$
$$b_1 = -K_p' + K_i' - 2K_d'$$
$$b_2 = K_d'$$

そして、この場合の z領域の伝達関

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

これを一般的な形式と比較すると、PIDはCNTL_2P2Zコントロールの特殊なケースにすぎないことが分かります。 この場合、次のようになります。

$$a_1 = -1 \text{ および } a_2 = 0$$

電圧ループの場合は、これらのP、I、Dの各係数は Pgain、Igain、Dgainとなります。これらのP、I、Dの各係数は、Q26の形式で使用されます。GUI環境(またはCCSのウォッチ・ビュー)からのチューニングを簡素化するために、これら3つの係数をさらにスケーリングして0～999(Pgain_Gui、Igain_Gui、Dgain_Gui)の範囲の値にします。

GUI環境から、2極(fp1, fp2)、2零点(fz1, fz2)、ゲイン(Kdc)を使用して電圧ループをチューニングすることもできます。これらのパラメータにより、b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiの各係数がI5Q10形式で提供され、2P2Zコントローラ用に5つのQ26係数に変換されます。推奨はされませんが、ウォッチ・ビューを使用して、b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiの各値を直接CCS環境から変更することも可能です。これらの計算の詳細については、HVPSFB-Calculations.xlsファイルを参照してください。極、零点、ゲイン、スイッチング周波数を基にして係数値を導出するための式も、GUIソース・ファイルから簡単に探すことができます。

実行中に係数を容易に切り替える機能が提供されているため、このプロジェクトでは、ループ・チューニングの2つの方法を容易に評価できるようになっています。切り替えは、GUI上の2P2Z(On)/PID(Off)ボタンをクリックするか、CCSからウォッチ・ビュー上でpid2p2z_GUI変数を0または1に変更するだけで行えます。GUI環境からのPIDベースのループ・チューニング(pid2p2z_GUI = 0)は、開始点として使用されました。さらに、PIDでチューニングされたこれらの係数に対応する極、零点、ゲインが、2番目の方法(pid2p2z_GUI = 1)に基づいた追加のループ・チューニングの開始点として使用されました。その後、極、零点、ゲインをGUI環境から変更して最適な動的性能にチューニングすることにより、大幅に向上した結果が実現されました。デフォルトでは、これらのチューニングされた極、零点、ゲインの各値に基づいた係数(pid2p2z_GUI = 1 – デフォルト)が使用されます。

注: 2-pole 2-zeroのコントローラの係数の補正値(adjustments)を使用してシステムをチューニングする際に、極、零点、Kdcの各値の選択をGUIで行った結果、任意の係数(b2、b1、b0、a2、a1)の大きさが32以上になった場合は、GUIではこれらの係数値をコントローラに送信しません。

定電流(CC)と定電力(CP)

このプロジェクトには、定電流と定電力の機能(functionality)を試すためのシンプルな実装が含まれています。デフォルトでは、この機能はディセーブルになっています。定電流機能は、電圧ループ・コントローラのクランプ値を変更することで実装されます。定電力機能は、負荷を基にした電圧ループ基準コマンドを補正して、出力で定電力を保持することで実装されます。これらの機能の完全なソフトウェア・フローチャートを次に示します。

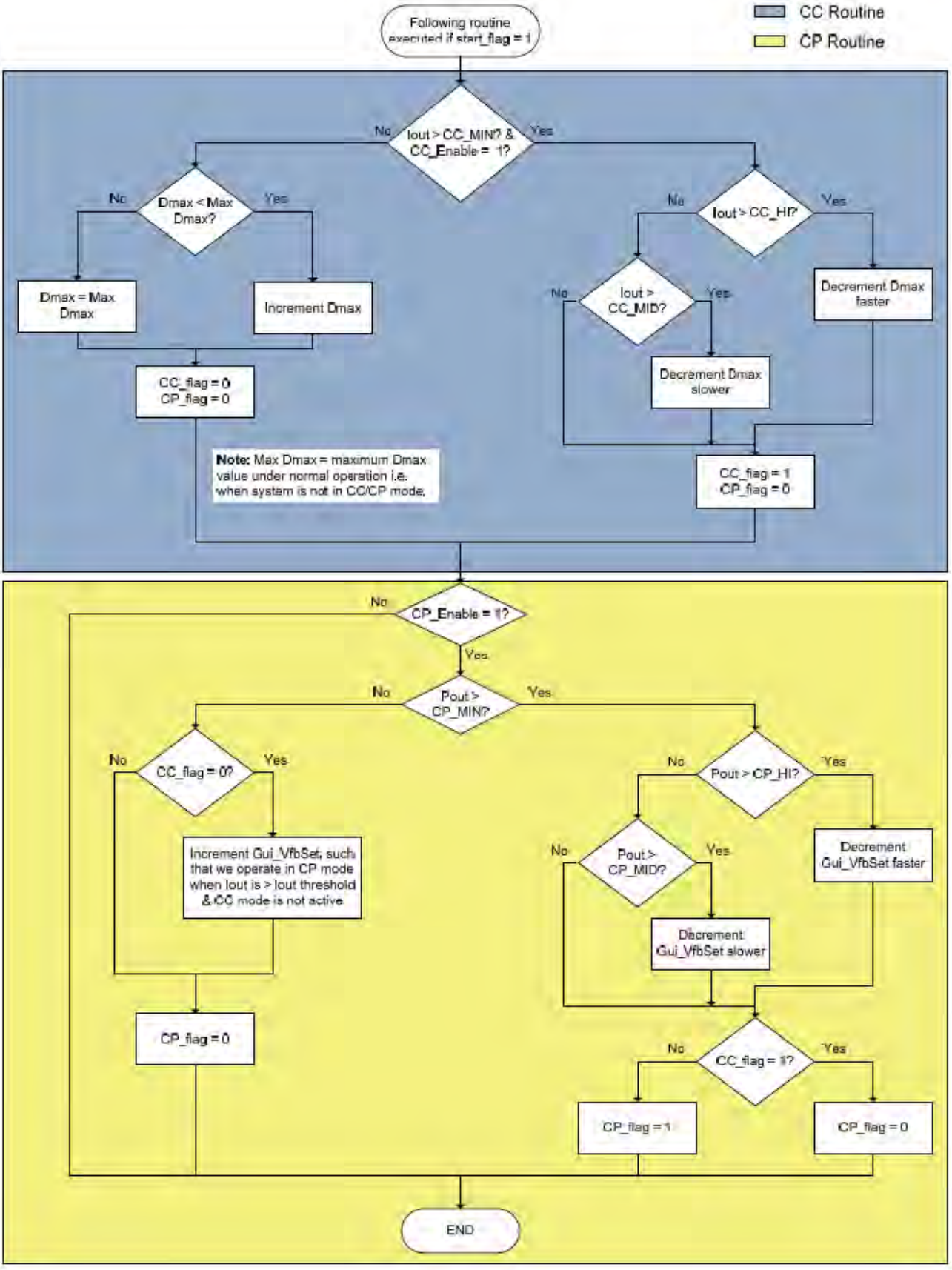


図21 定電流および定電力ソフトウェア・フローチャート

➢ 手順(Procedure)

ビルドと負荷のプロジェクト(Build and Load Project)

事前に構成された作業環境(work environment)を使用してこのビルドを短時間で実行するには、次のステップに従います。

1-2.ビルド1のステップ1、2とまったく同様のステップに従ってください。直前にCCSを使用した際にビルド1で作業を行っていた場合は、このプロジェクトでも同じワークスペース(作業領域)が開くはずです。そうでない場合は、File -> Switch Workspace(ワークスペースの切り替え)をクリックした後に検索して、ビルド1で使用していたワークスペースを開くことができます。ワークスペースが保存されなかったか、削除された場合は、ビルド1のステップ3、4とまったく同様のステップに従ってください。

3. メイン・ファイル内のビルド2固有の初期化コードを探して調べます。このコードではすべての制御ブロックが構成・初期化され、制御フローに接続されます。
4. HVPSFB-Settings.h内の2を、インクリメンタル・ビルド・オプションとして選択します。

注: HVPSFB-Settings.h内のインクリメンタル・ビルド・オプションを変更した場合は必ず、「Rebuild All(すべてをリビルドする)」を行ってください。

5. Project ->「Rebuild All(すべてをリビルドする)」ボタンをクリックして、ツールがビルド・ウィンドウで実行されるのを確認します。
6. Target ->「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。プロジェクト構成に応じて、プログラムがFLASHメモリまたはRAMメモリにロードされます。このプロジェクトは、F2802x_FLASHの構成でのみ提供されます。これで、Main() を開始する準備ができました。

デバッグ環境のウィンドウ(Debug Environment Windows)

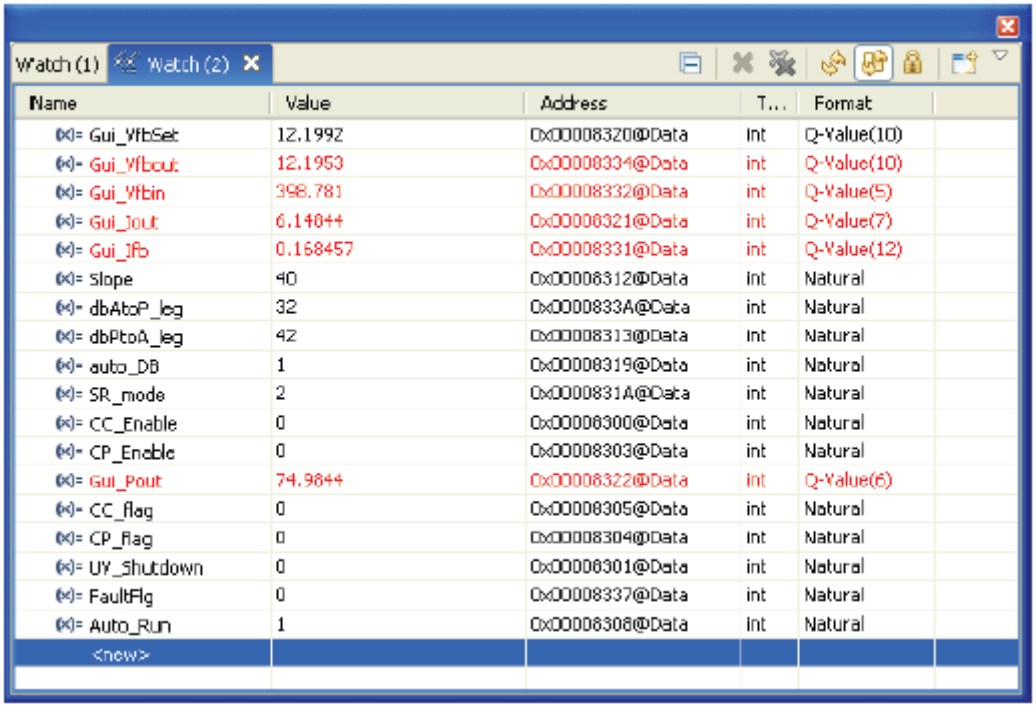
コードのデバッグ中には、ローカル変数とグローバル変数を監視するのが一般的です。Code Composer Studioでこの監視を行うには、メモリ・ビューやウォッチ・ビューなど様々な手段があります。さらに、Code Composer Studioには時間領域(および周波数領域)のプロットの作成機能があります。これにより、グラフ・ウィンドウを使用して波形を見ることも可能です。

7. デバッグ環境が起動してもウォッチ・ビューが開かない場合は、新規のウォッチ・ビューを開き、次の手順に従って各種パラメータを追加してください。

   メニュー・バーで、View(表示) -> Watchをクリックします。

   「Watch (1)」タブをクリックします。ウォッチ・ビューがすでに、ビルド1用に保存されたデバッグ環境から開かれている場合は、ウォッチ・ビューの  ボタンをクリックします。「Watch (2)」タブが開いたら、それをドラッグして選択したウィンドウで表示することが可能になります。

   ウォッチ・ビューに任意の変数を追加できます。「Name」カラムの空白のボックスに監視対象の変数のシンボル名を入力し、キーボードからエンター・キーを押します。必要に応じて、「Format」を修正してください。ウォッチ・ビューは次のような外観になっています。この時点では初期化されていない変数もメイン・コード内に存在するため、変数に不要なデータが含まれている可能性があることに注意してください。



ウォッチ・ビューの追加の変数を確認してください。

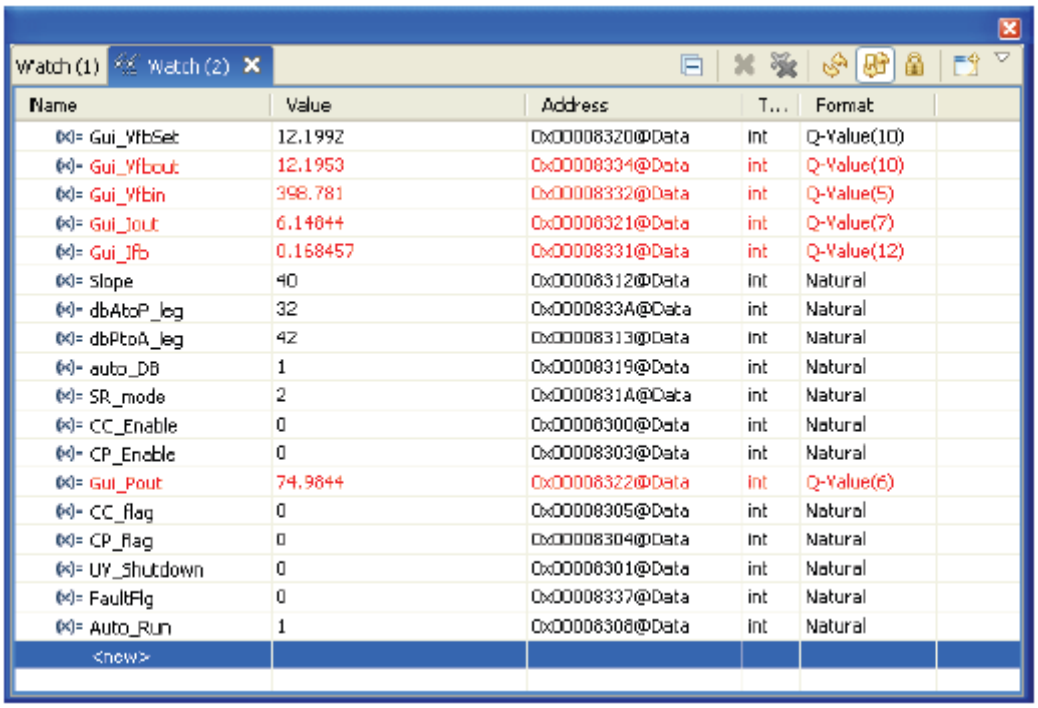
8. Gui_VfbSetは、出力電圧コマンドの設定に使用されます。

コードの実行

9. ビルド1の手順のステップ17～21を行って、リアル・タイム・モードと、ウォッチ・ビューの継続的なリフレッシュ、および必要な場合はウォッチ・ビューの 継続的なリフレッシュ間隔 の変更をイネーブルにします。
10. <F8>キー、ツールバーのRunボタン、メニュー・バーのTarget-> Runのどれかを使用して、コードを実行します。
11. 適切な抵抗負荷をPSFBシステムのDC出力側に取り付けます。出力12Vで約3A～6Aの電流を引き出す負荷は、良い開始点となります。

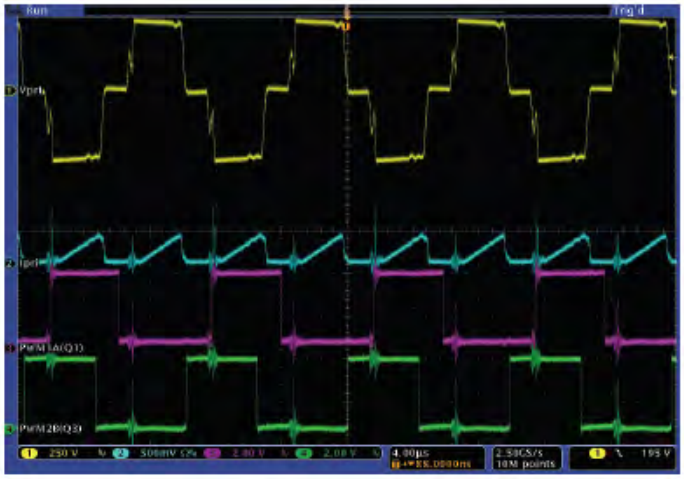
注: 安全上の理由から、絶縁DC電源を使用してDC入力400Vをボードに供給することが推奨されます。

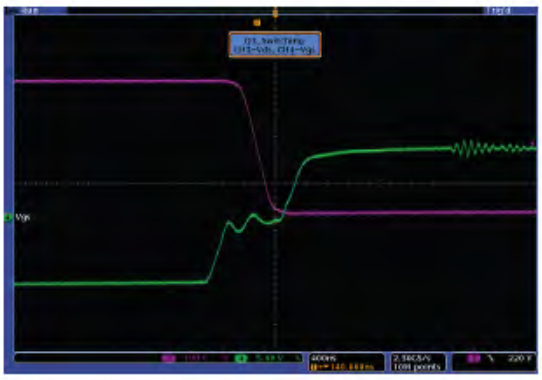
12. J1、J2の入力に、400VのDCで電力を供給します。
13. デフォルトでは、Auto_Runは1に設定されます。そうでない場合は、ウォッチ・ビューからAuto_Runを1にしてください。出力電圧はこの時点で、最大12Vまで傾き始めます。この出力電圧の上昇率(ramp up rate)は、変数VfbSlewRateを変更することで変更できます。
14. これは、約400Vの入力電圧と約6A出力の負荷を使用する、出力側で12.2Vのシステムの動作に対応するウォッチ・ビューです。

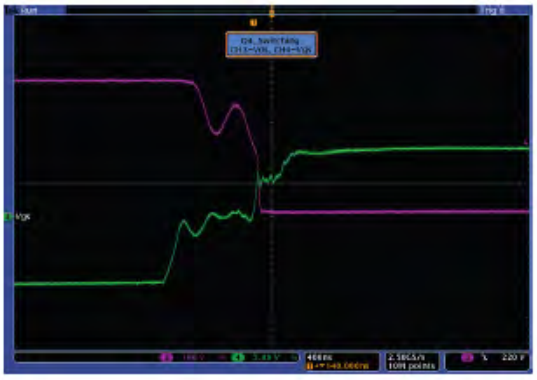


Name	Value	Address	T...	Format
Gui_VfbSet	12.1992	0x00008320@Data	int	Q-Value(10)
Gui_Vfbout	12.1953	0x00008334@Data	int	Q-Value(10)
Gui_Vfbin	398.781	0x00008332@Data	int	Q-Value(5)
Gui_Iout	6.14844	0x00008321@Data	int	Q-Value(7)
Gui_Ifb	0.168457	0x00008331@Data	int	Q-Value(12)
Slope	40	0x00008312@Data	int	Natural
dbAtoP_leg	32	0x0000833A@Data	int	Natural
dbPtoA_leg	42	0x00008313@Data	int	Natural
auto_DB	1	0x00008319@Data	int	Natural
SR_mode	2	0x0000831A@Data	int	Natural
CC_Enable	0	0x00008300@Data	int	Natural
CP_Enable	0	0x00008303@Data	int	Natural
Gui_Pout	74.9844	0x00008322@Data	int	Q-Value(6)
CC_flag	0	0x00008305@Data	int	Natural
CP_flag	0	0x00008304@Data	int	Natural
UV_Shutdown	0	0x00008301@Data	int	Natural
FaultFlg	0	0x00008337@Data	int	Natural
Auto_Run	1	0x00008308@Data	int	Natural
<new>				

15. 次に示すオシロスコープのキャプチャでは、上記で説明された条件下で見られるトランスの1次電圧、検知された1次電流(primary sensed current)、2つの対角線上で向き合ったスイッチ(Q1とQ3)を駆動するPWM波形です。スイッチQ3のZVSスイッチングとスイッチQ4のLVSスイッチングも、これらの条件下で示されます。







16. デフォルトでは、同期整流器はモード2で駆動されます。同期整流器の動作のモードは、ウォッチ・ビューからSR_mode変数を0、1、2に変更することで変更できます。各種SRモードでの、引き込まれる入力電流の量の変化と出力電圧の変化を観察してください。同期整流器のスイッチを駆動するPWM波形を精査することもできます。(3Aを下回る)非常に低い負荷で動作している場合は、各種SRモードを変更しないでください。このような例では、デフォルトのSRモード2を使用してください。

17. 出力電圧上と入力電流上で負荷*を様々に変化させた場合の影響を観察します。これらは事実上、出力電圧に影響しないはずです。同様に、入力電圧*を変化させた場合の影響も観察してください。この場合も、出力電圧には実質的に何の影響もないはずです。

注: これらの変更が、このドキュメントの仕様値のセクションに記載されているボードの機能の範囲内で行われるようにしてください。

18. オシロスコープを使用して、PWMゲート駆動信号、入力電圧と入力電流、出力電圧等の様々な波形を精査することも可能です。この絶縁DC-DCコンバータについて、上記のような高電圧大電流を精査する際には、適切な安全対策を取り、適切なグラウンド要件について考慮する必要があります。

19. このプロジェクトには、「定電流(CC)」と「定電力(CP)」の機能(functionality)を試すためのシンプルな実装が含まれていて、それらの対応するフラグ(CC_EnableとCP_Enable)をイネーブル/ディセーブルすることで、これらを使用した実験を行うことが可能です。

20. リアルタイム・モードのMCUを完全に停止するには、2ステップのプロセスを実行します。DC入力400Vをオフにして数秒待機します。最初に、ツールバーのHaltボタンか Target -> Halt を使用してプロセッサを停止します。次に、  ボタンを再度クリックしてMCUをリアルタイム・モードから解除してから、MCUをリセットします。

21. CCSを閉じます。

演習の終了

5 ソフトウェアの概要 - VMC

3.1 ソフトウェア制御フロー

HVPSFB_VMCプロジェクトでは、「Cバックグラウンド(C-background)/ASM-ISR」フレームワークを利用します。この方法ではアプリケーションの主な支援プログラムとしてCコードを利用し、すべてのシステム管理タスク、意志決定(decision making)、インテリジェンス(intelligence)、ホストとの通信(host interaction)を担当します。アセンブリ・コードは厳密にISR(割り込みサービス・ルーチンInterrupt Service Routine)に限定されます。ISRではすべてのクリティカルな制御コードを実行します。通常、このクリティカルな制御コードにはADC読み取り(reading)、制御計算、PWMとDACのアップデート等が含まれます。図22に、このプロジェクトの一般的なソフトウェア・フローを示します。

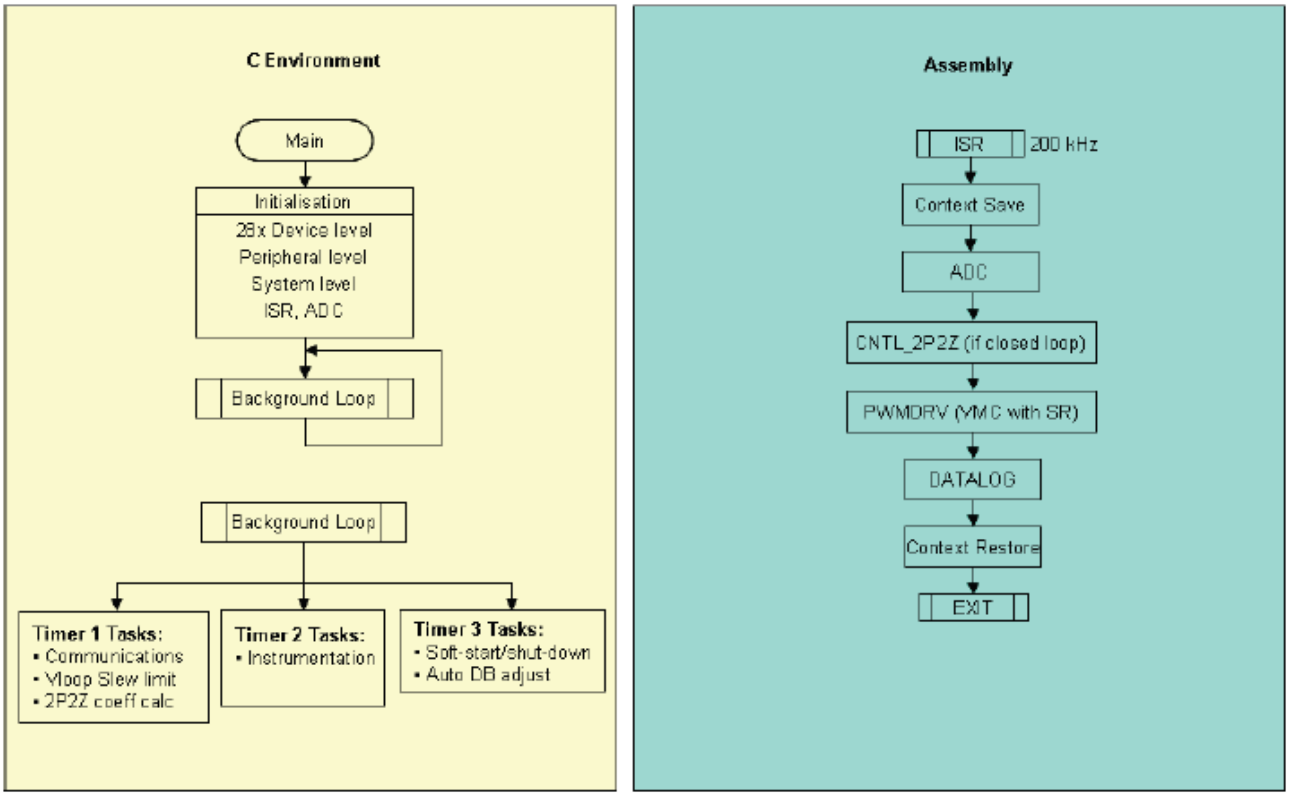


図22 VMCのソフトウェア・フロー

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ファイルは次の通りです。

HVPSFB-Main.c – アプリケーションの背後で「中枢部」となるファイルであり、アプリケーションの初期化、実行、管理に使用されます。

HVPSFB-DevInit.c – マイクロコントローラの一回限りの初期化と構成を担当します。このファイルには、クロック、PLL、GPIOのセットアップ等の関数などが入っています。

ISRは、次のファイルのみで構成されます。

HVPSFB-DPL-ISR.asm – このファイルには、すべてのコードの中で最もクリティカルな(all time critical)「制御タイプ」コードが入っています。このファイルには(一度だけ実行される)初期化セクションと、ランタイム・セクション自体をトリガするために使用されるPWMタイムベースと同じ速度で(通常は)実行されるランタイム・セクションが入っています。

パワー・ライブラリ関数(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とアセンブリ両方のコンポーネントが含まれていることがあります。このプロジェクトでは、次に挙げるライブラリ・モジュールが使用されます。Cのモジュール名と、それに対応するアセンブリのモジュール名は次の通りです。

表 5 ライブラリ・モジュール

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

制御ブロックは、次のように図解することも可能です (図23)。

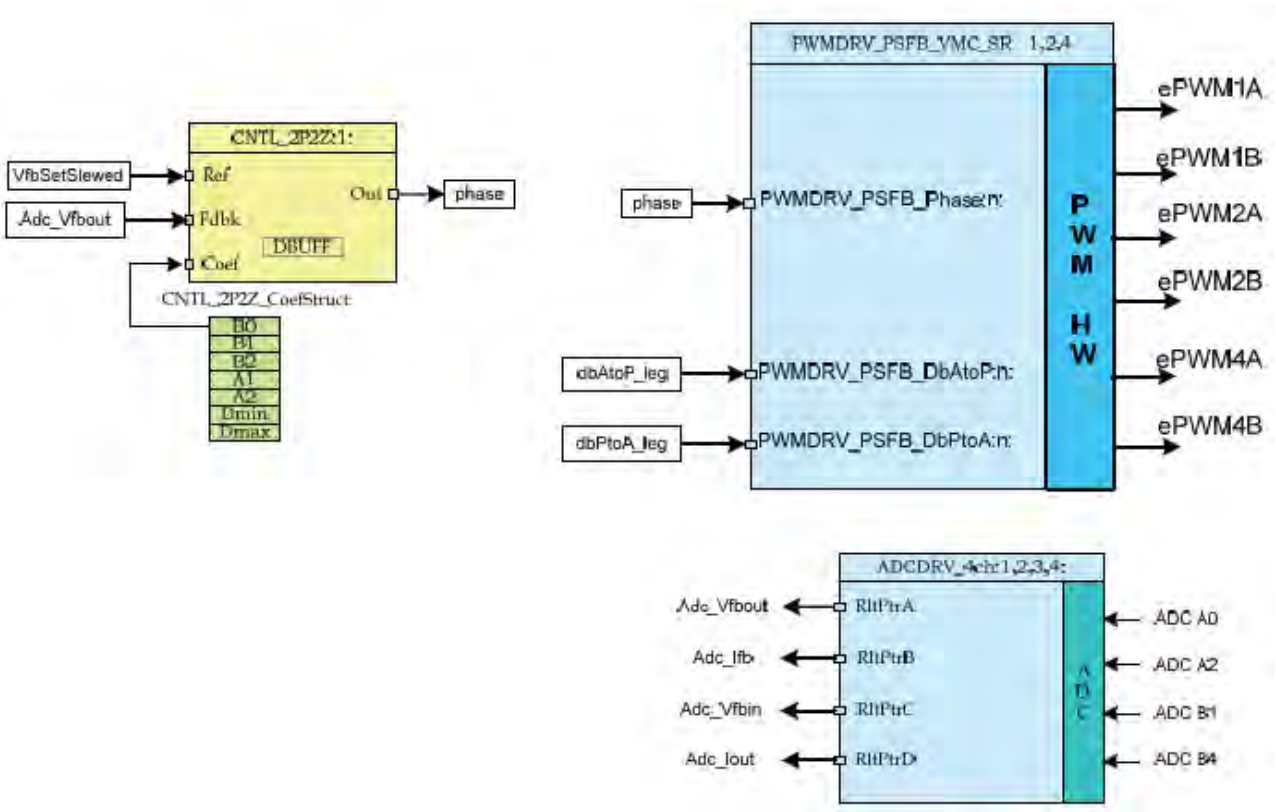


図23 VMC ソフトウェアのブロック

図23のモジュールのカラー・コーディングに注目してください。「濃い目の青」のブロックはC2000マイクロコントローラのハードウェア・モジュールを表しています。「オレンジ色」のブロックは、これらのモジュールのソフトウェア・ドライバです。「黄色」のブロックは制御ループのコントローラ・ブロックです。ここでは2-pole 2-zeroのコントローラを使用していますが、3-pole 3-zeroやPI/PID等、このアプリケーションに合うように実装できればどのコントローラでも問題ありません。図17のようなモジュール・ライブラリ構造体(modular library structure)を使用して、システム・ソフトウェア・フロー全体を視覚化し、理解しやすくしたものが図24になります。このような図により、各種機能を使用したり、追加/削除したりすることも容易になります。この事実は、インクリメンタル・ビルド・アプローチ(incremental build approach)を実装することにより、このプロジェクトで十分に証明されています。これは、次のセクションでさらに詳細に説明されます。

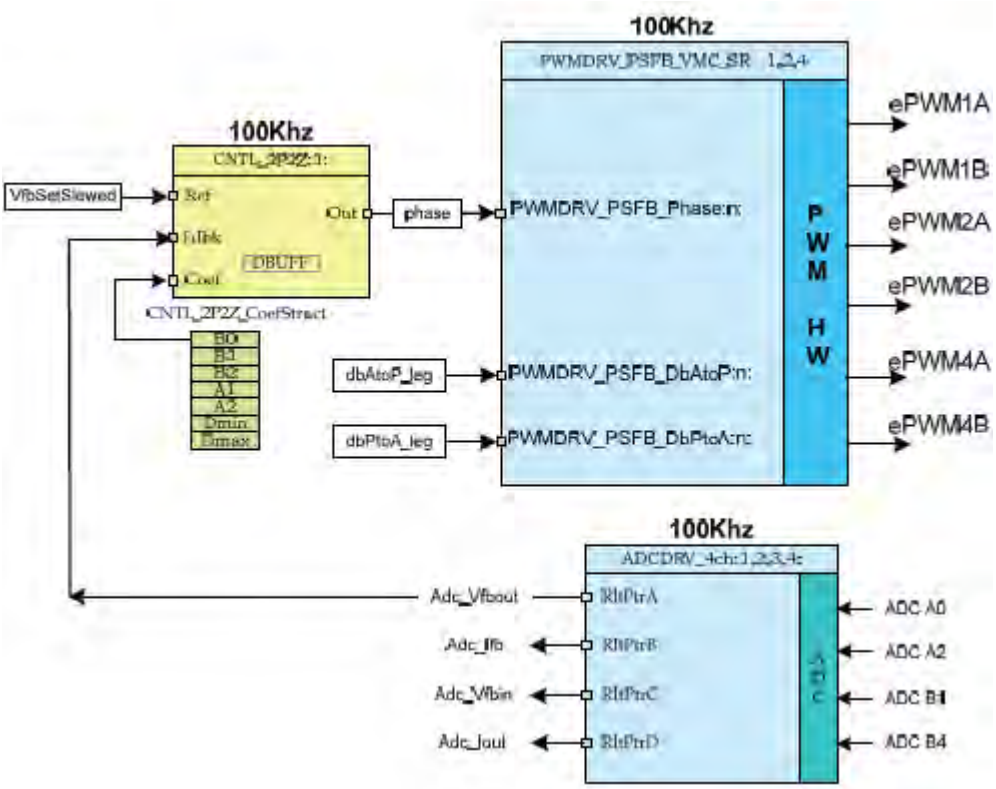


図 24 PCMC制御のフロー

システムは、ひとつの電圧フィードバック・ループにより制御されます。図24には、各制御ブロックが実行される速度(rate)も示されています。例えば、電圧ループ・コントローラは100kHz(PWMスイッチング周波数と同じ)という速度で実行されます。次に、上記で実装された制御について説明します。

検知された出力電圧(Adc_Vfbout)は、電圧コントローラ内の基準電圧コマンド(Vref)をスルーレートで補正したもの(slewed version)(VfbSetSlewed)と比較されます。電圧コントローラの出力では、フルブリッジの2つのレッグを駆動するPWM信号間の位相シフトを直接制御します。これにより、フルブリッジの2つのレッグ間の位相重複の量が検出され、出力電圧が安定化します。dbAtoP_leg値とdbPtoA_leg値では、フルブリッジの「能動 – 受動(Active to Passive)」レッグと「受動– 能動(Passive to Active)」レッグにそれぞれデッド・バンド値を提供します。これらの値は、負荷範囲全体でZVS/LVSを実現するために使用されます。

5.2 インクリメンタル・ビルド

このプロジェクトは、2つのインクリメンタル・ビルドに分かれています。このことにより、ボードとソフトウェアについて学習し、慣れることが容易になっています。このアプローチはボードのデバッグ/テストにも有効です。

ビルドのオプション(選択肢)を次に示します。特定のビルド・オプションを選択するには、HVPSFB-Settings.h ファイルにあるマクロINCR_BUILDを、下の表に記載された対応するビルド選択項目に設定します。ビルド・オプションを選択した後、rebuild-all(すべてをリビルドする)のコンパイラ・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。次の章では、各ビルド・オプションの実行についてさらに詳細に説明します。

インクリメンタル・ビルドのオプション	
INCR_BUILD = 1	ADCのフィードバックを使用した開ループPSFBの駆動 (PWM駆動回路と検知回路をチェック)
INCR_BUILD = 2	閉電圧ループ(VMCモードの完全なPSFB)

表 6 VMCのインクリメンタル・ビルド・オプション

6 インクリメンタル・ビルドを実行するための手順

PSFBシステムを作成するためのCフレームワーク用のメイン・ソース・ファイル、ISRアセンブリ・ファイル、プロジェクト・ファイルは、次に示すディレクトリにあります(最新のソフトウェア・パッケージを使用してください。2012年3月付のバージョン1.1が最新のソフトウェア・バージョンです)。

..\controlSUITE\development_kits\HVPSFB_v1.1\HVPSFB_VMC

このソフトウェアに入っているプロジェクトは、CCSv4をターゲットとしています。

警告
ボード上は高電圧状態になっているため、ラボ環境の経験を積んだ電源の専門家以外はボードを取り扱わないようにしてください。このボードを安全に評価するには、絶縁処理を施した適切な高電圧DC電源を使用する必要があります。ボードにDC電力が印加される前に、電圧計と適切な抵抗負荷または電気的負荷(resistive or electronic load)を出力に取り付けてください。電力が印加されている間は、絶対にユニットに触れないでください。

次に示すステップに従って、HVPSFB_VMCソフトウェアに含まれている例をビルドして実行してください。

6.1 ビルド 1: ADC のフィードバックを使用した開ループのチェック

➢ 目的

このビルドの目的は、システムの開ループの動作を評価し、PWMおよびADCドライバ・モジュールを検証し、ボード上のMOSFETドライバ回路と検知回路を検証し、Code Composer Studio (CCS)の動作に慣れることです。このシステムは開ループで動作するため、ADCで計測された値はこのビルドでは計測(instrumentation)の目的でのみ使用されます。プロジェクトのビルドと実行に必要なステップを以降で紹介していきます。

➢ 概要

ビルド1のソフトウェアは、オシロスコープで出力波形を見て、位相をCCS上でインタラクティブに補正することで発生する出力電圧の位相の変化の影響を観察することにより、ユーザーが位相シフト・フルブリッジPWMドライバ・モジュールを短時間で評価できるように構成されています。また、ADCでサンプリングされたデータをウォッチ・ビューで見ることで、ユーザー側でADCドライバ・モジュールを評価することが可能です。

前章で述べたように、PWMドライバとADCドライバのマクロのインスタンス化は_DPL_ISR内部で実行されます。図25は、このビルドで使用するブロックです。ePWM1AおよびePWM1BではQ2とQ3のフルブリッジ・スイッチを駆動しますが、ePWM2AとePWM2BではQ1とQ4のフルブリッジ・スイッチを駆動します。ePWM4AとePWM4BではQ6とQ5の同期整流器スイッチを駆動します。

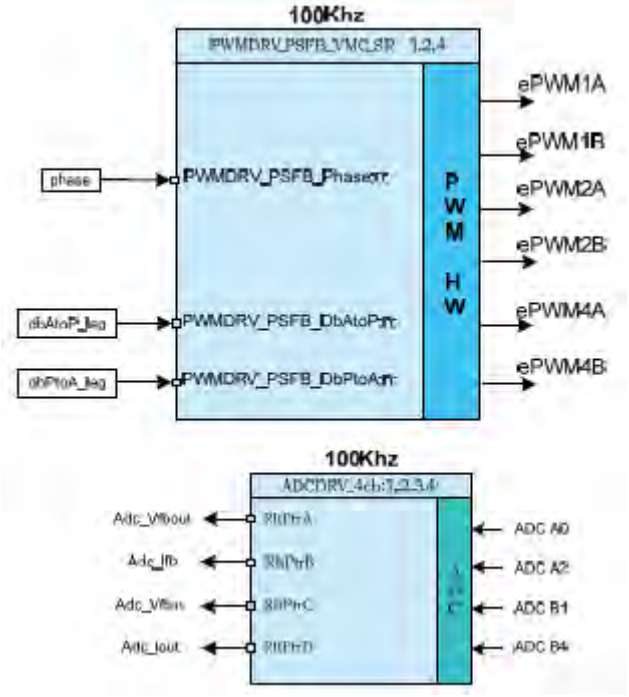


図25 ビルド1のソフトウェア・ブロック

これらのPWM信号は100kHzの周波数で、つまり10usという周期で生成される必要があります。60MHzで動作するMCUを使用する場合は、ePWM1、ePWM2、ePWM4のどれかのタイムベース・カウンタの1カウントが16.667nsに対応します。このことが意味するのは、PWM周期10usがタイムベース・カウンタ(TBCNT1、TBCNT2、TBCNT4)の600カウントと等価であるということです。ePWM1モジュールとePWM2モジュールはアップカウント・モードで動作するように構成されますが、ePWM4はアップダウン・カウント・モードで動作します。ePWM1Aの出力とePWM1Bの出力はデューティ・サイクル50%で動作し、互いに相補的となっています。同様に、ePWM2AとePWM2Bはデューティ・サイクル50%で動作し、互いに相補的です。ePWM2のタイムベースの位相は、ePWM1の位相に関連して動的に変更可能です。これらのPWM波形を図14に示します。

PWMドライバ・モジュールに*phase*を入力すると、PWM1タイムベースとPWM2タイムベース間の位相シフト量が決まります。この*phase*の値により、フルブリッジの対角線上で向き合ったスイッチのペアを駆動するPWM信号間の重複の量が制御されます。*phase*が増大すると、重複の量が増加し、それにより二次側に伝達されるエネルギーの量が増加します。TBPHS2の値は、入力された「*phase*」コマンドから導出されます。

表7に、599というTBPRD 値について導出されたTBPHS2の値の例を記載します。

表 7位相の参考値

位相(Q24)	TBPHS2 = (位相*TBPRD/2^25)	位相シフト(単位: 度)
2097152d	37	22.5
8388608d	149	90
16776704d	299	180

フルブリッジの対角線上のスイッチの各ペアが、1PWM周期に1回重複することに注意してください。このことが意味するのは、大部分の重複が、位相シフトが180度に近づいた時に発生するということです。

アセンブリのISRは、ePWM1のZRO (TBCNT1 = 0) イベント時にトリガされます。この場合には、制御ドライバ・マクロが実行され、TBPHS2レジスタとTBPHS4レジスタがアップデートされます。

ADCの入力がサンプリングされる場所に関しては、重要な考慮事項があります。ADC入力信号の品質(integrity)は非常に重要ですが、それはADCが、アナログ領域からの信号とデジタル領域からの信号の接点になる(interface)ためです。電力段でスイッチをオン/オフにすると、信号上である程度のノイズや外乱(disturbance)が発生し、この時点の前後で検知される結果になる可能性があります。ADCの入力にこのノイズが出現しないように、上記の信号にはフィルタリング処理が施されますが、それ以前に、この外乱を回避できる時点でADCの入力をサンプリングする方が賢明です。

また、セクション2.5で説明したように、検知された出力電圧は可能な限り、出力電圧値が平均値に近くなっているスイッチング・サイクルの適切な時点でサンプリングする必要があります。これを実現するために、ADCの入力信号は、可能な限りノイズの少ないサンプルが得られ、かつ平均出力電圧のサンプリングも行えるような時間にサンプリングされます。フルブリッジの場合、このことは、2つの対角線上のスイッチ間の重複(ここでは、重複とは両方のスイッチが同時にオンになる時間を示します)の中間点で、つまりMOSFETスイッチングから可能な限り離れている時点でサンプリングを行うことにより実現されます。これにより、ADC変換の結果にスイッチング・ノイズが反映されないようにすることが可能になります。C2000デバイスのADCモジュールとPWMモジュールの柔軟性により、ADC変換の非常に高精度で柔軟性の高いトリガリングが可能になっています。ADCドライバのモジュールは、12ビットADCの変換結果を読み取り、Q24の値に変換するために使用されます。1PWMサイクルごとに、PWM2 SOCA (変換Aの開始)を使用して5回分のADC変換がトリガされます。

保護機能(Protection)

この段階で、このプロジェクトで使用されているシャットダウン機構を紹介することが適切と思われます。ボードには、オンチップのアナログ・コンパレータ1を使用した、トランスの1次電流用の過電流保護機能が実装されています。基準トリップ・レベルは、内蔵の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を構成して、検知された電流が設定された限界値より大きい場合は常に、ePWM1上で1回限りの(ワンショット)トリップ・アクションを生成するようにします。C2000デバイス上のトリップ機構の柔軟性により、様々なトリップ・イベント時に様々なアクションを実行することが可能です。このプロジェクトでは、ePWM1A、ePWM1B、ePWM2A、ePWM2Bの各出力が即座にLowに駆動されることで、電力段(power stage)が保護されます。デバイスのリセットが実行されるまで、これらの出力はこの状態(ステート)に保持されます。これらの計算の詳細については、HVPSFB-Calculations.xlsファイルを参照してください。定電流モードまたは定電力モードで動作している場合には、出力低電圧シャットダウン機構もソフトウェアから実装されます。

入力低電圧および入力過電圧ロックアウトも、ソフトウェア的に実装されます。

リソースのマッピング(割り当て)

C2000マイクロコントローラ～HVPSFB段間の主要な信号接続を、次の表にまとめてあります。PWMのマッピングがVMCプロジェクトとPCMCプロジェクトでは異なることに注意してください。コントローラ・カード上のジャンパ(J2、J3 – PCMCおよびVMCのPWM駆動ジャンパのイネーブル)を、VMCモード用に正しく構成する必要があります(デフォルトのジャンパ位置は、PCMC動作用にセットされています)。2つのモード用のジャンパ構成を次に紹介します。

- **PCMC**: J2(1) -> J3(1)、J2(2) -> J3(2) 、J2(3) -> J3(3) 、J2(4) -> J3(4) 、J2(7) -> J3(7) 、J2(8) -> J3(8) 、
- **VMC**: J2(1) -> J3(3) 、J2(2) -> J3(4) 、 J2(3) -> J3(1) 、 J2(4) -> J3(2) 、 J2(7) -> J3(8) 、 J2(8) -> J3(7)

表 8 HVPSFB信号インターフェイスの参考表(リファレンス) – VMC

信号名	説明	C2000コントローラへの接続
ePWM-1A	PWM駆動(フルブリッジ・スイッチQ2用)	GPIO-00
ePWM-1B	PWM駆動(フルブリッジ・スイッチQ3用)	GPIO-01
ePWM-2A	PWM駆動(フルブリッジ・スイッチQ1用)	GPIO-02
ePWM-2B	PWM駆動(フルブリッジ・スイッチQ4用)	GPIO-03
ePWM-4A	PWM駆動(同期整流器スイッチ Q6用)	GPIO-06
ePWM-4B	PWM駆動(同期整流器スイッチQ5用)	GPIO-07
Vout	PSFB出力電圧	ADC-A0
Ifb	トランスの1次電流	ADC-A2/COMP1A
Ifb	トランスの1次電流	ADC-A4/COMP2A*
Vfbin	PSFBの入力電圧	ADC-B1
Iout1	PSFBの出力電流	ADC-B3
Iout2	PSFBの出力電流(大幅にフィルタ処理)	ADC-B4

* コントローラ・カード上のデフォルトのジャンパJ4構成です。入力はジャンパJ4を選択することで構成可能です。

注: HVPSFB_PCMCプロジェクトとHVPSFB_VMCプロジェクトでは、device_supportディレクトリに配置されているファイルではなく、それら自体のプロジェクト・ディレクトリに配置されているDSP2802x_Comp.hファイルとDSP2802x_EPWM.h ヘッダ・ファイルを使用します。device_support ディレクトリ内の2つのファイルは、後でControlSuiteがアップデートされる際にアップデートされます。アップデート時には、これらのアップデートされたファイルを2つのプロジェクトに使用できます。

➢ 手順(Procedure)

CCSを起動してプロジェクトを開く

このビルドを短時間で実行するには、次のステップに従います。

1. ベース・ボード上でジャンパJ6が装着されておらず、J8が装着されていることを確認します。
2. デフォルトでは、コントローラ・カードのPiccolo Macro上の抵抗R6とジャンパJ1を外して、FLASHからの起動をイネーブルにします。RAMの実行とプログラミング、またはFLASHのプログラミングを行うには、この2つを再度装着(Re-populate)します。USBコネクタを、エミュレーション用のPiccoloコントローラ・カードに接続します。DC約400Vを出力するためにセットされた、適切な絶縁処理済みのDC電源のご使用が推奨されます。DC電源がメイン・ボード上のJ1とJ2に接続される前には、DC電源をオフ

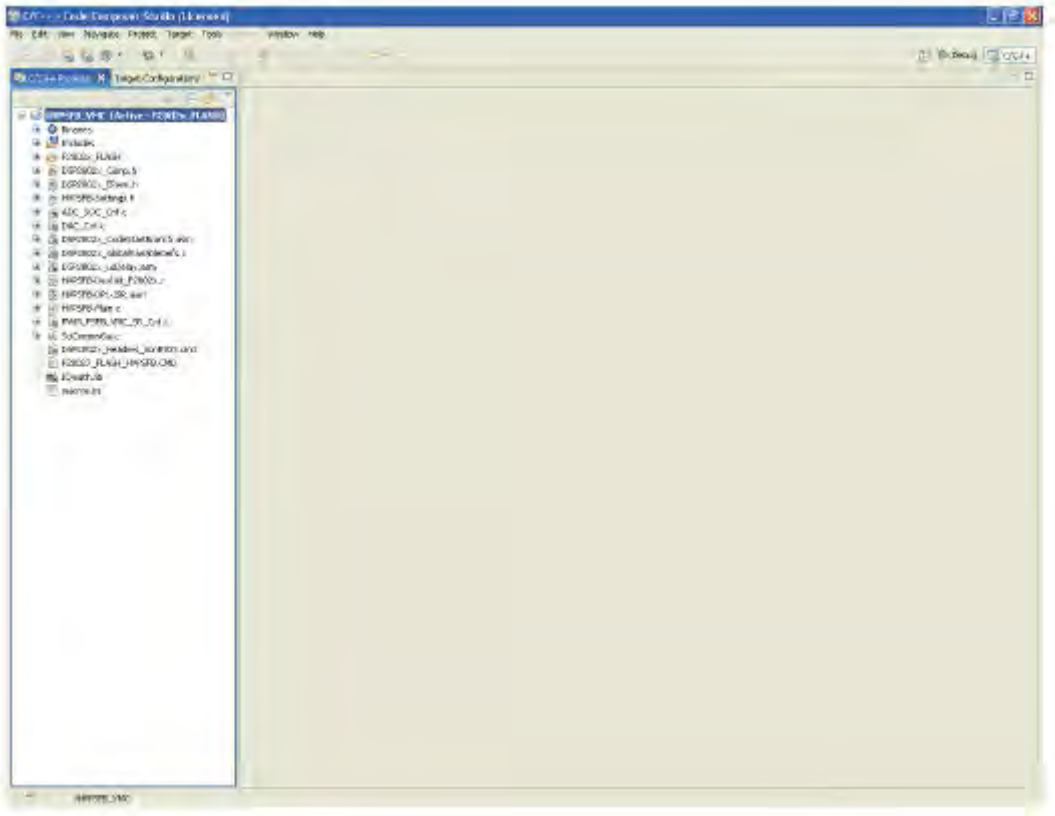
のままにしておく必要があります。20AWGの配線600Vを使用して、電源をJ1とJ2に接続します。この接続の極性が正しいことを確認してください。適切な抵抗負荷/DC電気的負荷を、位相シフト・フルブリッジ・システムのDC出力側のJ3とJ4に取り付けます。この時点ではDC電源400Vをオンにしないでください。TP1～TP2間のバイアス電源を、DC約11V(この電圧は12Vより低くする必要があります)を使用して起動します。

3. デスクトップのCode Composer Studioアイコンをダブルクリックします。Code Composer Studioを最大表示にして、お使いのディスプレイ全面に表示します。Welcome 画面が開いた場合は閉じてください。

4. プロジェクトには、MCUハードウェア上で動作させることのできる実行可能な出力ファイル(.out)の開発に必要なすべてのファイルとビルド・オプションが入っています。メニュー・バー上で、「Project」 -> 「Import Existing CCS/CCE Eclipse Project (既存のCCS/CCE Eclipseプロジェクトをインポートする)」をクリックして、「Select root directory (ルート・ディレクトリの選択)」から「..\controlSUITE\development_kits\HVPSFB_v1.1\HVPSFB_PCMC」ディレクトリを探して選択します。「Projects」タブ下で、HVPSFB_PCMCにチェックが入っていることを確認してください。「Finish」をクリックします。

   このプロジェクトでは、プロジェクトをビルドするために必要なすべてのツール(コンパイラ、アセンブラ、リンカ)を呼び出します。

5. 左側のプロジェクト・ウィンドウで、プロジェクトの左側にあるプラス記号(+)をクリックします。お使いのプロジェクト・ウィンドウが次のように表示されます。



Device Initialization (デバイスの初期化)ファイル、Main(メイン)ファイル、ISRファイル

注: ソース・ファイルは絶対に変更せず、内容を見て調べるだけにしてください。

6. プロジェクト・ウィンドウのファイル名をダブルクリックして、HVPSFB-DevInit_F2802x.c を開いて調べてください。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、ペリフェラル・クロックのイネーブルがセットアップされていることを確認した後、共有のGPIOピンが構成されていることを確認してください。

7. HVPSFB-Main.cを開いて調べてください。DeviceInit()関数に対して行われた呼び出しと、その他の変数初期化を確認してください。また、各種インクリメンタル・ビルド・オプション(具体的には現時点でコンパイルしようとしているビルド)用のコード、ISR初期化、(;;) ループ用のバックグラウンドも確認してください。

8. メイン・ファイル内で、ビルド1の固有の初期化コードの下にある、次に示すコードを探して調べてください。このコードでは、制御フロー内でPWMDRV_PSFB_VMC_SRブロックが接続され、初期化されます。

```
    PWMDRV_PSFB_VMC_SR_CNF(1, PWM_PRD, 1, 1);    // ePWM1 and ePWM2, Period=PWM_PRD,
                                                 //SR_Enable=1 |ePWM4|, Comp1_Prot=1
// Connect the PWMDRV_PSFB_VMC_SR driver block
    PWMDRV_PSFB_Phase1  = &phase;           // Point to the phase net
    PWMDRV_PSFB_DbAtoP1 = &dbAtoP_leg;      // Point to the left leg dead band adjust
    PWMDRV_PSFB_DbPtoA1 = &dbPtoA_leg;      // Point to the right leg dead band adjust
```

9. メイン・ファイル内で、ビルド1の固有の初期化コードの下にある、次に示すコードを探して調べてください。このコードでは、制御フロー内でADCDRV_4CHブロックが構成され、初期化され、接続されます。

```
#define     Vfb_outR    AdcResult.ADCRESULT1    //
#define     IfbR        AdcResult.ADCRESULT2    //
#define     Vfb_inR     AdcResult.ADCRESULT3    //

#define     IoutR       AdcResult.ADCRESULT4    //

// Channel Selection for Cascaded Sequencer
    ChSel[0] = 0;               // A0 - O/P Voltage - Dummy
    ChSel[1] = 0;               // A0 - O/P Voltage
    ChSel[2] = 2;               // A2 - Transformer Primary Current
    ChSel[3] = 9;               // B1 - I/P Voltage
    ChSel[4] = 12;              // B4 - Iout2

    TrigSel[0] = 7;             // O/P Voltage sampling triggered by EPWM2 SOCA - Dummy
    TrigSel[1] = 7;             // O/P Voltage sampling triggered by EPWM2 SOCA
    TrigSel[2] = 7;             // Transformer Primary Current sampling triggered by EPWM2 SOCA
    TrigSel[3] = 7;             // I/P Voltage sampling triggered by EPWM2 SOCA
    TrigSel[4] = 7;             // Iout sampling triggered by EPWM2 SOCA

    EALLOW;
    AdcRegs.SOCPRICTL.bit.SOCPRIORITY = 4; // SOC0-3 are high priority
    EDIS;
   ADC_SOC_CNF(ChSel,TrigSel,ACQPS, 16, 0);// ACQPS=8, No ADC channel triggers an interrupt IntChSel > 15,
                                           // Mode= Start/Stop (0)
// ADC feedback connections
    ADCDRV_4ch_RltPtrA = &Adc_Vfbout;
    ADCDRV_4ch_RltPtrB = &Adc_Ifb;
    ADCDRV_4ch_RltPtrC = &Adc_Vfbin;
    ADCDRV_4ch_RltPtrD = &Adc_Iout;
```

10. HVPSFB-DPL-ISR.asmを開いて調べてください。_DPL_Initと _DPL_ISRのセクションを確認してください。ここでは、PWMドライバとADCドライバのマクロのインスタンス化が、初期化用とランタイム用にそれぞれ行われます。オプションとして、調べたファイルを閉じることもできます。

プロジェクトのビルドとロード(Build and Load the Project)

11. HVPSFB-Settings.h ファイルの1、つまりインクリメンタル・ビルド・オプションを選択します。

注: HVPSFB-Settings.hのインクリメンタル・ビルド・オプションを変更した場合は必ず、「Rebuild All(すべてをリビルドする)」を行ってください。

12. Project ->「Rebuild All(すべてをリビルドする)」ボタンをクリックして、ツールがビルド・ウィンドウで実行されるのを確認します。

13. Target ->「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。まだターゲット構成ファイルが選択されていない場合は、新規ターゲット構成ファイルを開くように指示するCCSのメッセージが表示されます。有効なターゲット構成ファイルがこの接続用に生成済みの場合は、ステップ15に進んでください。新規ターゲット構成ウィンドウで「.ccxml」という拡張子の付いたファイル名(例: xds100-F28027.ccxml)を入力して、今回の作業を行うターゲットの名前とします。「Use shared location(共有の場所を使用)」をチェックして、Finish(終了)をクリックします。

14. .ccxmlファイルを開いて、Connection(接続)として「Texas Instruments XDS100v2 USB Emulator」をクリックし、そのデバイス名の下でスクロールダウンして「TMS320F28027」を選択します。「 Save(保存)」をクリックします。

15. Target -> 「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。選択されたプロジェクト構成に応じて、プログラムがFLASHメモリまたはRAMメモリにロードされます。このプロジェクトは、F2802x_FLASHの構成でのみ提供されます。これで、Main() を開始する準備ができました。

デバッグ環境のウィンドウ(Debug Environment Windows)

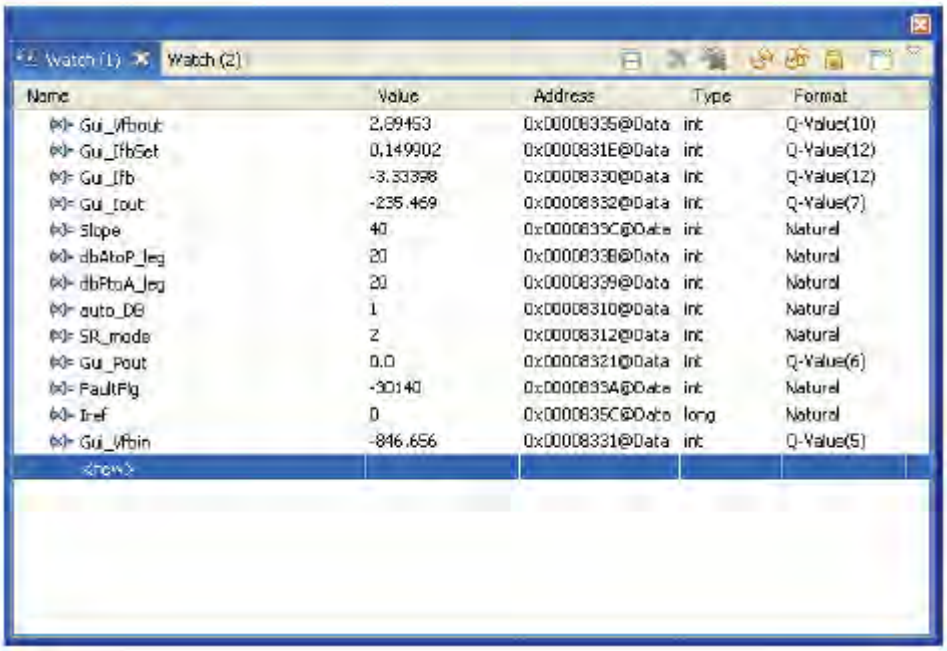
コードのデバッグ中には、ローカル変数とグローバル変数を監視するのが一般的です。Code Composer Studioでこの監視を行うには、メモリ・ビューやウォッチ・ビューなど様々な手段があります。さらに、Code Composer Studioには時間領域(および周波数領域)のプロットの作成機能があります。これにより、グラフ・ウィンドウを使用して波形を見ることも可能です。

16. デバッグ環境が起動してもウォッチ・ビューが開かない場合は、新規のウォッチ・ビューを開き、次の手順に従って各種パラメータを追加してください。

    メニュー・バーで、View(表示) -> Watchをクリックします。

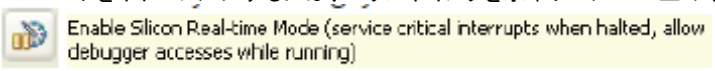
    「Watch (1)」タブをクリックします。ウォッチ・ビューに任意の変数を追加できます。「Name」カラムの空白のボックスに監視対象の変数のシンボル名を入力し、キーボードからエンター・キーを押します。必要に応じて、「Format」を修正してください。ウォッチ・ビューは次のような外観になっています。この時点では初期化されていない変数もメイン・コード内に存在するため、変数に不要なデータが含まれている可能性があることに注意してください。

    (設定されている場合)FaultFlgは(上記で説明された)過電流条件を示しており、この条件によりPWM出力がシャットダウンされます。デバイスのリセット(ステップ32に従って正しく行うこと)まで、PWM出力はこのステートに保持されます。Ipri_trip変数により、オンチップ・コンパレータ1用に内部10ビットDAC基準レベルがセットされます。これがQ15の数であることに注意してください。



リアルタイム・エミュレーションを使用する

リアルタイム・エミュレーションは、*MCUの実行中に*Code Composer Studio内のウィンドウのリフレッシュ・レートを最大10Hzにアップデートすることを可能にする、特別なエミュレーション機能です。これにより、グラフやウォッチ・ビューがアップデート可能になるだけでなく、ユーザーがウォッチ・ウィンドウやメモリ・ウィンドウの値を変更して、MCUの動作にその変更を反映できるようになります。これは、制御のロウ・パラメータ(raw parameters)をオンザフライで調整する場合などに非常に役立ちます。

17. リアルタイム・モードをイネーブルにするには、マウスポインタを水平ツールバー上のボタンの上に置き、次のボタンをクリックします。 

18. メッセージ・ボックスが表示されることがあります。その場合は「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」にセットされます。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」にセットされると、メモリ値とレジスタ値がホスト・プロセッサに渡されて、デバッガのウィンドウがアップデートできるようになります。

19. ここで、同じ水平ツールバー上の次のボタンをクリックします。


20. 多数のウィンドウが開いた場合、エミュレーション・リンクに関連する帯域幅に制限があるため、継続的なリフレッシュでアップデートされるウィンドウと変数が多くなりすぎて、リフレッシュ周波数が完全に停止する可能性があります。ウォッチ・ビューのボタン を右クリックして、「Customize Continuous Refresh Interval..(継続的なリフレッシュ間隔をカスタマイズする)」を選択します。「Continuous refresh interval(milliseconds) (継続的なリフレッシュ間隔(ミリ秒単位))」の値を変更して、ウォッチ・ビューの変数のリフレッシュ・レートを下げます。通常では、4000msのリフレッシュ・レートであれば、このアプリケーション・ノートの演習用としては十分です。

21. ウォッチ・ビュー用に、「 Continuous Refresh(継続的なリフレッシュ)」ボタン  をクリックします。

コードの実行

22. <F8>キー、ツールバーのRunボタン、メニュー・バーのTarget-> Runのどれかを使用して、コードを実行します。

23. ウォッチ・ビューで、変数*phase*を0.015625 (Q24)に設定する必要があります。この変数は、位相シフト・コマンドをPWMDRV_PSFB_VMC_SRモジュールに対して示します。*phase*には、0.005より小さい値を使用しないでください。

24. 適切な抵抗負荷をPSFBシステムのDC出力側に取り付けます。出力12Vで約3A～6Aの電流を引き出す負荷は、良い開始点となります。

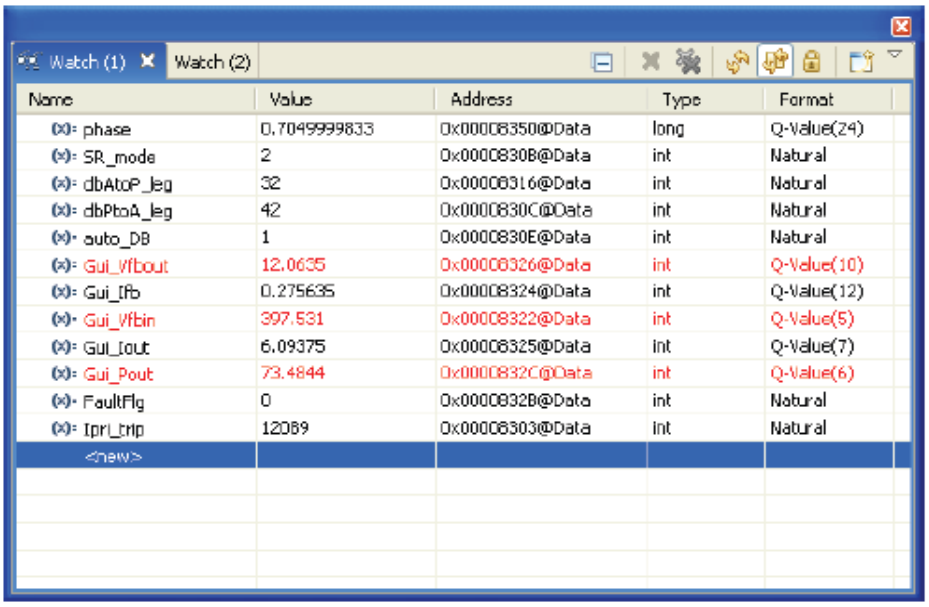
注: 安全上の理由から、絶縁DC電源を使用してDC入力400Vをボードに供給することが推奨されます。

25. J1、J2の入力に、400VのDCで電力を供給します。

26. ウォッチ・ビューで*phase*を前より高い値(例: 0.1)に設定して、位相コマンドを大きくします。出力電圧を注意して観察し、ボードの許容範囲を超えないようにしてください。ある一定の*phase*値を使用して動作している場合は、負荷が急に小さくなると出力電圧が上昇することを覚えていてください。**したがって、ビルド1での動作中には、負荷を急激に変化させたり、*phase* コマンドの数値を大幅に上げたりしないでください。**

27.ウォッチ・ビューの様々なADC結果の、様々な*phase*値を観察します。

28.入力電圧が約400Vで、出力12Vでの負荷が約6Aの、0.705 (Q24)という値の*phase*コマンドを使用するシステムの動作に対応するウォッチ・ビューを次に示します。



29. 次に示すオシロスコープのキャプチャは、上記で説明された条件下で見られるトランスの1次電圧と、検知された1次電流を示しています。

30. デフォルトでは、同期整流器はモード2で駆動されます。同期整流器の動作のモードは、ウォッチ・ビューからSR_mode変数を0、1、2に変えることで変更できます。各種SRモードでの、引き込まれる入力電流の量の変化と出力電圧の変化を観察してください。同期整流器のスイッチを駆動するPWM波形を精査することもできます。非常に低い負荷で動作しているか、出力電圧が非常に低い(6Vを下回る)場合は、各種SRモードを変更しないでください。このような例では、デフォルトのSRモード2を使用してください。

31. 様々な*phase*値を試して、対応するADC結果を観察したり、少しずつ*phase*を増加させたりします。出力電圧を常に注意して観察し、ボードの許容範囲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、入力電圧と入力電流、出力電圧等の様々な波形を精査することも可能です。この絶縁DC-DCコンバータについて、上記のような高電圧大電流を精査する際には、適切な安全対策を取り、適切なグラウンド要件について考慮する必要があります。

32. リアルタイム・モードのMCUを完全に停止するには、2ステップのプロセスを実行します。DC入力400Vをオフにして数秒待機します。最初に、ツールバーのHaltボタンか Target -> Halt を使用してプロセッサを停止します。次に、  ボタンを再度クリックしてMCUをリアルタイム・モードから解除してから、MCUをリセットします。

33. Code Composer Studioは、次の演習用に動作させておくことも、任意で閉じることもできます。

演習の終了

6.2 ビルド 2: 閉電圧ループ(VMC モードの完全な HVPSFB)

➢ 目的

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

➢ 概要

図26は、このビルドで使用するソフトウェア・ブロックです。PWMドライバとADCドライバのブロックは、前述のビルドと同じように使用されます。電圧ループには2-pole 2-zeroのコントローラが使用されています。アプリケーションの制御ループの要件によっては、他のコントローラ・ブロック(3-pole 3-zeroやPI等)の使用も可能です。図に示すように、電圧ループ・ブロックは100KHzで実行されます。CNTL2P2Zは、IIRフィルタ構造体により実現される2次補償回路(2nd order compensator)です。この関数(機能)はどのペリフェラルからも独立しているため、CNF関数の呼び出しは必要としません。

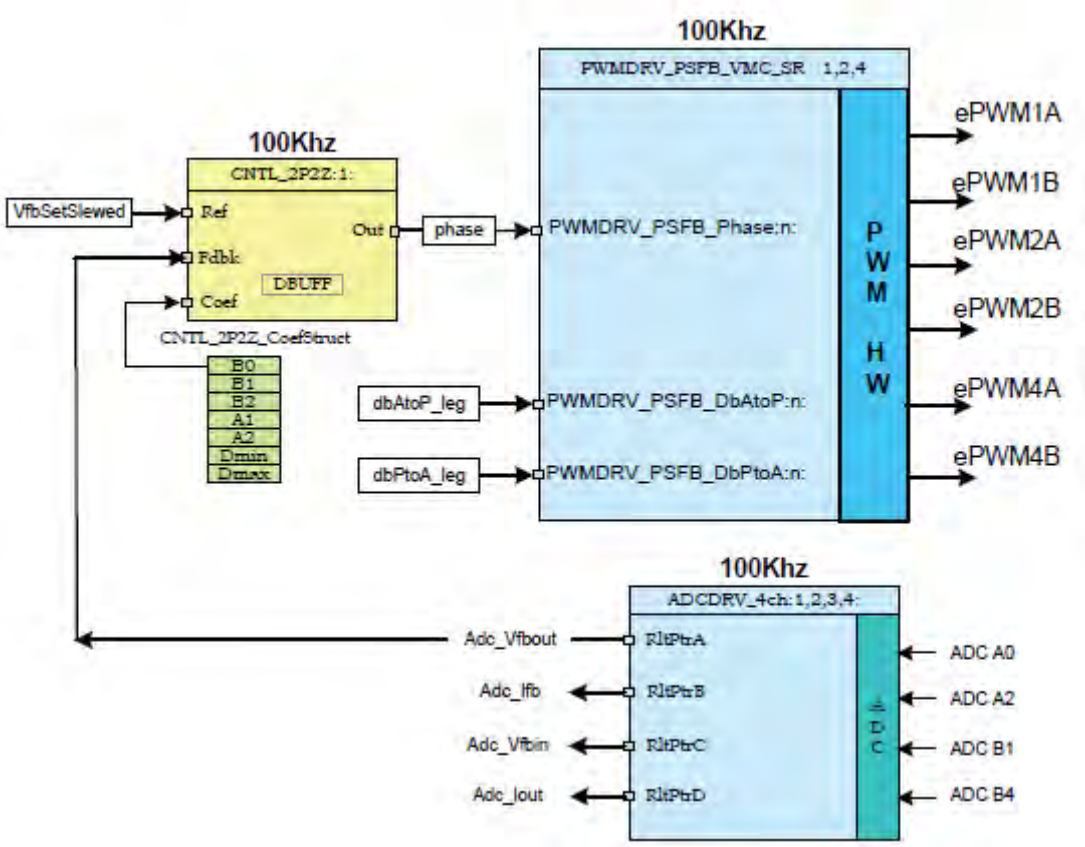


図 26 ビルド2のソフトウェア・ブロック

修正する必要のある5つの係数(coefficients)が、構造体CNTL_2P2Z_CoefStruct1のエレメントとして格納されています。この構造体の他のエレメントは、コントローラ出力をクランプ(制限)するために使用されます。システムで複数のループが必要な場合は、CNTL_2P2Zブロックを複数回インスタンス化(instantiate)できます。各インスタンスでは別個の係数セットを持つことが可能です。5つの係数を試行錯誤しながら個別に直接扱うことはほぼ不可能であるため、数学的な解析(数理解析)および/またはmatlabやmathcad等のツールによる支援が必要になります。これらのツールではボード線図、根軌跡(root-locus)等の機能を使用して、位相マージン、ゲイン・マージン等を決定できます。

ループ・チューニングをシンプルに保ち、複雑な数学ツールや解析ツールを使用せずにおくために、より直観的に理解しやすいP、I、Dという係数ゲイン(coefficient gains)をB0、B1、B2、A1、A2に効率的にマッピングすることにより、係数選択の問題の自由度のレベルが5段階から3段階に縮小されました。これにより、P、I、Dが別個かつ段階的に補正できるようになっています。このマッピングの式を次に示します。

補償回路ブロック(CNTL_2P2Z)には2つの極と2つの零点があり、汎用IIRフィルタの構造体がベースとなっています。また、基準入力とフィードバック入力もあります。電圧ループの場合は、検知された出力電圧(Adc_Vfbout)がフィードバックになりますが、コントローラに

対する基準入力は出力基準電圧コマンド(Vref)をスルーレートで補正したもの(slewed version) (VfbSetSlewed) になります。伝達関数は次の式で求められます。

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

PIDコントローラ の再帰形式(recursive form)は、次の差分方程式(difference equation)により求められます。

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

ここで、次のようになります。

$$b_0 = K_p' + K_i' + K_d'$$
$$b_1 = -K_p' + K_i' - 2K_d'$$
$$b_2 = K_d'$$

そして、この場合の z領域の伝達関数形式 は次のようになります。

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

これを一般的な形式と比較すると、PIDはCNTL_2P2Zコントロールの特殊なケースにすぎないことが分かります。 この場合、次のようになります。

$$a_1 = -1 \text{ および } a_2 = 0$$

電圧ループの場合は、これらのP、I、Dの各係数は Pgain、Igain、Dgainとなります。これらのP、I、Dの各係数は、Q26の形式で使用されます。GUI環境(またはCCSのウォッチ・ビュー)からのチューニングを簡素化するために、これら3つの係数をさらにスケーリングして0～999(Pgain_Gui、Igain_Gui、Dgain_Gui)の範囲の値にします。

GUI環境から、2極(fp1, fp2)、2零点(fz1, fz2)、ゲイン(Kdc)を使用して電圧ループをチューニングすることもできます。これらのパラメータにより、b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiの各係数がI5Q10形式で提供され、2P2Zコントローラ用に5つのQ26係数に変換されます。推奨はされませんが、ウォッチ・ビューを使用して、b2_Gui、b1_Gui、b0_Gui、a2_Gui、a1_Guiの各値を直接CCS環境から変更することも可能です。これらの計算の詳細については、HVPSFB-Calculations.xlsファイルを参照してください。極、零点、ゲイン、スイッチング周波数を基にして係数値を導出するための式も、GUIソース・ファイルから簡単に探すことができます。

実行中に係数を容易に切り替える機能が提供されているため、このプロジェクトでは、ループ・チューニングの2つの方法を容易に評価できるようになっています。切り替えは、GUI上の2P2Z(On)/PID(Off)ボタンをクリックするか、CCSからウォッチ・ビュー上でpid2p2z_GUI変数を0または1に変更するだけで行えます。GUI環境からのPIDベースのループ・チューニング(pid2p2z_GUI = 0)は、開始点として使用されました。さらに、PIDでチューニングされたこれらの係数に対応する極、零点、ゲインが、2番目の方法(pid2p2z_GUI = 1)に基づいた追加のループ・チューニングの開始点として使用されました。その後、極、零点、ゲインをGUI環境から変更して最適な動的性能にチューニングすることにより、大幅に向上した結果が実現されました。デフォルトでは、これらのチューニングされた極、零点、ゲインの各値に基づいた係数(pid2p2z_GUI = 1 – デフォルト)が使用されます。

注: 2-pole 2-zeroのコントローラの係数の補正値(adjustments)を使用してシステムをチューニングする際に、極、零点、Kdcの各値の選択をGUIで行った結果、任意の係数(b2、b1、b0、a2、a1)の大きさが32以上になった場合は、GUIではこれらの係数値をコントローラに送信しません。

➢ 手順(Procedure)

ビルドと負荷のプロジェクト(Build and Load Project)

事前に構成された作業環境(work environment)を使用してこのビルドを短時間で実行するには、次のステップに従います。

1-2.ビルド1のステップ1、2とまったく同様のステップに従ってください。直前にCCSを使用した際にビルド1で作業を行っていた場合は、このプロジェクトでも同じワークスペース(作業領域)が開くはずです。そうでない場合は、File -> Switch Workspace(ワークスペースの切り替え)をクリックした後に検索して、ビルド1で使用していたワークスペースを開くことができます。ワークスペースが保存されなかったか、削除された場合は、ビルド1のステップ3、4とまったく同様のステップに従ってください。

3. メイン・ファイル内のビルド2固有の初期化コードを探して調べます。このコードではすべての制御ブロックが構成・初期化され、制御フローに接続されます。

4. HVPSFB-Settings.h内の2を、インクリメンタル・ビルド・オプションとして選択します。

   注: HVPSFB-Settings.h内のインクリメンタル・ビルド・オプションを変更した場合は必ず、「Rebuild All(すべてをリビルドする)」を行ってください。

5. Project ->「Rebuild All(すべてをリビルドする)」ボタンをクリックして、ツールがビルド・ウィンドウで実行されるのを確認します。

6. Target ->「Debug Active Project(アクティブなプロジェクトをデバッグする)」をクリックします。プロジェクト構成に応じて、プログラムがFLASHメモリまたはRAMメモリにロードされます。このプロジェクトは、F2802x_FLASHの構成でのみ提供されます。これで、Main() を開始する準備ができました。

デバッグ環境のウィンドウ(Debug Environment Windows)

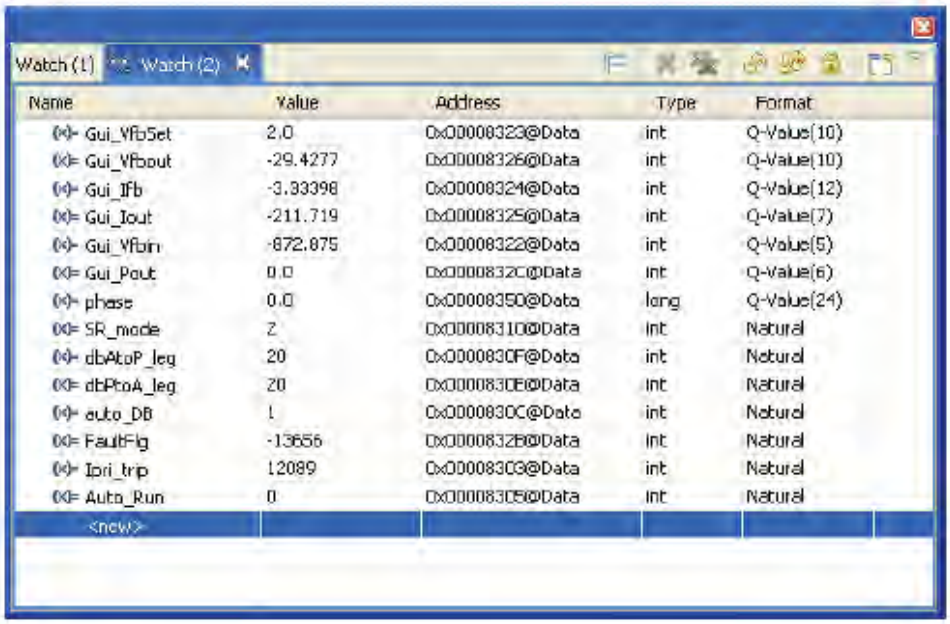
コードのデバッグ中には、ローカル変数とグローバル変数を監視するのが一般的です。Code Composer Studioでこの監視を行うには、メモリ・ビューやウォッチ・ビューなど様々な手段があります。さらに、Code Composer Studioには時間領域(および周波数領域)のプロットの作成機能があります。これにより、グラフ・ウィンドウを使用して波形を見ることも可能です。

7. デバッグ環境が起動してもウォッチ・ビューが開かない場合は、新規のウォッチ・ビューを開き、次の手順に従って各種パラメータを追加してください。

   メニュー・バーで、View(表示) -> Watchをクリックします。

   「Watch (1)」タブをクリックします。ウォッチ・ビューがすでに、ビルド1用に保存されたデバッグ環境から開かれている場合は、ウォッチ・ビューの New Watch View ボタンをクリックします。「Watch (2)」タブが開いたら、それをドラッグして選択したウィンドウで表示することが可能になります。

   ウォッチ・ビューに任意の変数を追加できます。「Name」カラムの空白のボックスに監視対象の変数のシンボル名を入力し、キーボードからエンター・キーを押します。必要に応じて、「Format」を修正してください。ウォッチ・ビューは次のような外観になっています。この時点では初期化されていない変数もメイン・コード内に存在するため、変数に不要なデータが含まれている可能性があることに注意してください。



ウォッチ・ビューの追加の変数を確認してください。

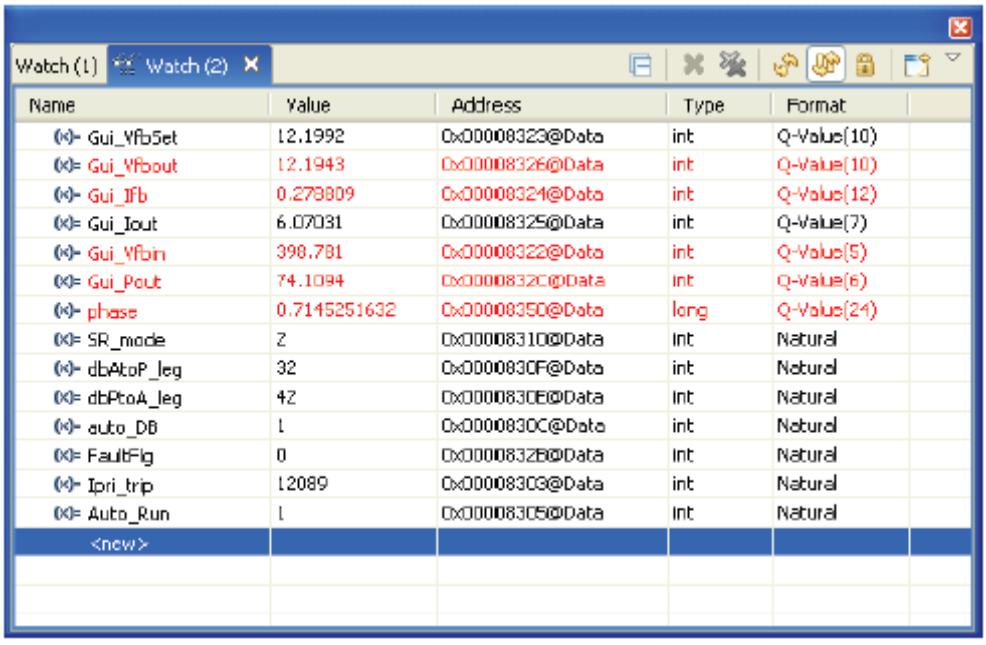
8. Gui_VfbSetは、出力電圧コマンドの設定に使用されます。

コードの実行

9. ビルド1の手順のステップ17～21を行って、リアル・タイム・モードと、ウォッチ・ビューの継続的なリフレッシュ、および必要な場合はウォッチ・ビューの継続的なリフレッシュ間隔 の変更をイネーブルにします。
10. <F8>キー、ツールバーのRunボタン、メニュー・バーのTarget-> Runのどれかを使用して、コードを実行します。
11. 適切な抵抗負荷をPSFBシステムのDC出力側に取り付けます。出力12Vで約3A～6Aの電流を引き出す負荷は、良い開始点となります。

注: 安全上の理由から、絶縁DC電源を使用してDC入力400Vをボードに供給することが推奨されます。

12. J1、J2の入力に、400VのDCで電力を供給します。
13. デフォルトでは、Auto_Runは1に設定されます。そうでない場合は、ウォッチ・ビューからAuto_Runを1にしてください。出力電圧はこの時点で、最大12Vまで傾き始めます。この出力電圧の上昇率(ramp up rate)は、変数VfbSlewRateを変更することで変更できます。
14. これは、約400Vの入力電圧と約**6A**出力の負荷を使用する、出力側で12Vのシステムの動作に対応するウォッチ・ビューです。



15. 次の写真は、上記で説明された条件下でキャプチャされた波形です。

16. デフォルトでは、同期整流器はモード2で駆動されます。同期整流器の動作のモードは、ウォッチ・ビューからSR_mode変数を0、1、2に変更することで変更できます。各種SRモードでの、引き込まれる入力電流の量の変化と出力電圧の変化を観察してください。同期整流器のスイッチを駆動するPWM波形を精査することもできます。(3Aを下回る)非常に低い負荷で動作している場合は、各種SRモードを変更しないでください。このような例では、デフォルトのSRモード2を使用してください。

17. 出力電圧上と入力電流上で負荷*を様々に変化させた場合の影響を観察します。これらは事実上、出力電圧に影響しないはずです。同様に、入力電圧*を変化させた場合の影響も観察してください。この場合も、出力電圧には実質的に何の影響もないはずです。

***注:これらの変更が、このドキュメントの仕様値のセクションに記載されているボードの機能の範囲内で行われるようにしてください。**

18. オシロスコープを使用して、PWMゲート駆動信号、入力電圧と入力電流、出力電圧等の様々な波形を精査することも可能です。この絶縁DC-DCコンバータについて、上記のような高電圧大電流を精査する際には、適切な安全対策を取り、適切なグラウンド要件について考慮する必要があります。

19. リアルタイム・モードのMCUを完全に停止するには、2ステップのプロセスを実行します。DC入力400Vをオフにして数秒待機します。最初に、ツールバーのHaltボタンか Target -> Halt を使用してプロセッサを停止します。次に、  ボタンを再度クリックしてMCUをリアルタイム・モードから解除してから、MCUをリセットします。

20. CCSを閉じます。

演習の終了

7 参考文献

このアプリケーション・ノートに記載されているよりもさらに詳細な情報については、次のガイドを参照してください。

1. UCC28950 600-W, Phase-Shifted, Full-Bridge Application Report (Literature Number SLUA560B) http://focus.ti.com.cn/cn/lit/an/slua560b/slua560b.pdf

2. 600-W, Phase-Shifted, Full-Bridge Converter (Literature Number SLUU421A) http://focus.ti.com/lit/ug/sluu421a/sluu421a.pdf

3. H. Nene, “Implementing Advanced Control Strategies for Phase Shifted Full-Bridge DC-DC Converters using Micro-Controllers” PCIM Europe 2011, Nuremberg, Germany.

4. HVPSFB-Calculations.xls – a spreadsheet showing key calculations for the HVPSFB project.
..\controlSUITE\development_kits\HVPSFB_v1.1\~Docs

5. QSG-HVPSB-Rev1.1.pdf– gives an overview on how to quickly run and experiment with the HVPSFB project using an intuitive GUI interface.
..\controlSUITE\development_kits\HVPSFB_v1.1\~Docs

6. HVPSFB_RevB-HWdevPkg – a folder containing various files related to the base board and Piccolo-A controller card.
..\controlSUITE\development_kits\HVPSFB_v1.1\~HVPSFB_HWdevPkg\RevB

7. F28xxx User’s Guides

http://www.ti.com/f28xuserguides

ご注意

Texas Instruments Incorporated 及びその関連会社（以下総称して TI といいます）は、最新の JESD46 に従いその半導体製品及びサービスを修正し、改善、改良、その他の変更をし、又は最新の JESD48 に従い製品の製造中止またはサービスの提供を中止する権利を留保します。お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての半導体製品は、ご注文の受諾の際に提示される TI の標準販売契約約款に従って販売されます。

TI は、その製品が、半導体製品に関する TI の標準販売契約約款に記載された保証条件に従い、販売時の仕様に対応した性能を有していることを保証します。検査及びその他の品質管理技法は、TI が当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、適用される法令によってそれ等の実行が義務づけられている場合を除き、必ずしも行なわれておりません。

TI は、製品のアプリケーションに関する支援又はお客様の製品の設計について責任を負うことはありません。TI 製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI 製部品を使用したお客様の製品及びアプリケーションに関連する危険を最小のものとするため、適切な設計上及び操作上の安全対策は、お客様にてお取り下さい。

TI は、TI の製品又はサービスが使用されている組み合せ、機械装置、又は方法に関連している TI の特許権、著作権、回路配置利用権、その他の TI の知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TI が第三者の製品もしくはサービスについて情報を提供することは、TI が当該製品又はサービスを使用することについてライセンスを与えるとか、保証又は是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない、又は TI の特許その他の知的財産権に基づき TI からライセンスを得て頂かなければならない場合もあります。

TI のデータ・ブック又はデータ・シートの中にある情報の重要な部分の複製は、その情報に一切の変更を加えること無く、且つその情報と関連する全ての保証、条件、制限及び通知と共になされる限りにおいてのみ許されるものとします。TI は、変更が加えられて文書化されたものについては一切責任を負いません。第三者の情報については、追加的な制約に服する可能性があります。

TI の製品又はサービスについて TI が提示したパラメーターと異なる、又は、それを超えてなされた説明で当該 TI 製品又はサービスを再販売することは、関連する TI 製品又はサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、且つ不公正で誤認を生じさせる行為です。TI は、そのような説明については何の義務も責任も負いません。

TI からのアプリケーションに関する情報提供又は支援の一切に拘わらず、お客様は、ご自身の製品及びご自身のアプリケーションにおける TI 製品の使用に関する法的責任、規制、及び安全に関する要求事項の全てにつき、これをご自身で遵守する責任があることを認め、且つそのことに同意します。お客様は、想定される不具合がもたらす危険な結果に対する安全対策を立案し実行し、不具合及びその帰結を監視し、害を及ぼす可能性のある不具合の可能性を低減し、及び、適切な治癒措置を講じるために必要な専門的知識の一切を自ら有することを表明し、保証します。お客様は、TI 製品を安全でないことが致命的となるアプリケーションに使用したことから生じる損害の一切につき、TI 及びその代表者にその全額の補償をするものとします。

TI 製品につき、安全に関連するアプリケーションを促進するために特に宣伝される場合があります。そのような製品については、TI が目的とするところは、適用される機能上の安全標準及び要求事項を満たしたお客様の最終製品につき、お客様が設計及び製造ができるようお手伝いをすることにあります。それにも拘わらず、当該 TI 製品については、前のパラグラフ記載の条件の適用を受けるものとします。

FDA クラス III（又は同様に安全でないことが致命的となるような医療機器）への TI 製品の使用は、TI とお客様双方の権限ある役員の間で、そのような使用を行う際について規定した特殊な契約書を締結した場合を除き、一切認められていません。

TI が軍需対応グレード品又は「強化プラスティック」製品として特に指定した製品のみが軍事用又は宇宙航空用アプリケーション、若しくは、軍事的環境又は航空宇宙環境にて使用されるように設計され、かつ使用されることを意図しています。お客様は、TI がそのように指定していない製品を軍事用又は航空宇宙用に使う場合は全てご自身の危険負担において行うこと、及び、そのような使用に関して必要とされるすべての法的要求事項及び規制上の要求事項につきご自身のみの責任により満足させることを認め、且つ同意します。

TI には、主に自動車用に使われることを目的として、ISO/TS 16949 の要求事項を満たしていると特別に指定した製品があります。当該指定を受けていない製品については、自動車用に使われるようには設計されてもいませんし、使用されることを意図しておりません。従いまして、前記指定品以外の TI 製品が当該要求事項を満たしていなかったことについては、TI はいかなる責任も負いません。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊／劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限 260℃以上の高温状態に、10 秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上