应用报告
ZHCA535–May 2013

数控高压 (HV) 太阳能最大功率点跟踪 (MPPT) 直流到直流 (DC-DC) 控制器，此控制器使用 C2000™ Piccolo™ 微控制器

Shamim Choudhury

摘要

这份应用报告介绍了一个数控 DC-DC 转换器的实施细节，此转换器被用作一个用于太阳能逆变器 (DC-AC) 应用的前端转换器。 为了利用一个 500W 太阳能电池板的最大容量，它执行一个支持最大功率点跟踪 (MPPT) 算法的隔离式 DC-DC 级。 它在 MPPT 算法生成的基准设定点上保持其输入电压，并且当一个下游 DC-AC 逆变器跨接在其输出上时，为其供电。 DC-AC 逆变器将电能从 DC-DC 级传输至被跨接在其自己输出上的一个仿真电网。 一个 C2000 Piccolo-B 控制卡和一个 500W 隔离式 DC-DC 级仿真模块 (EVM) 被用来执行整个 DC-DC 系统。

此 EVM 随 Piccolo-B 控制卡一起提供，而 Piccolo-A 卡不含此 EVM。 然而，一个 Piccolo-A 控制卡也可被用来执行 EVM 的完全控制。

内容

图片列表

图表列表

1 简介

基于太阳能的光伏 (PV) 系统提供一个环境友好电源。 这样的 PV 系统的一个关键特性是转换效率，在此效率上，电源转换器级可从 PV 阵列提取能量并传送给负载。 针对所有光照情况的 PV 输出的最大功率点跟踪 (MPPT) 可减少安装成本并大大增加 PV 电池板的功率输出。 因此，一个采用某些 MPPT 算法的 DC-DC 转换器通常被用作一个前端转换器来高效地提取 PV 输出功率并将 PV 输出电压转换为一个高压 DC 总线。 根据系统要求，DC-DC 转换器可用作一个隔离式功率级或一个非隔离式级。 然后，来自 DC-DC 转换器的高压被馈入，来为 DC-AC 逆变器供电，此逆变器最终为负载供电并连接至电网。

这个 C2000 MPPT DC-DC EVM 使用一个如图 1中所示的隔离式 DC-DC 级。 它包含 2 个 DC-DC 级：一个 2ph 交错式升压转换器和一个隔离式半桥 LLC 谐振转换器。

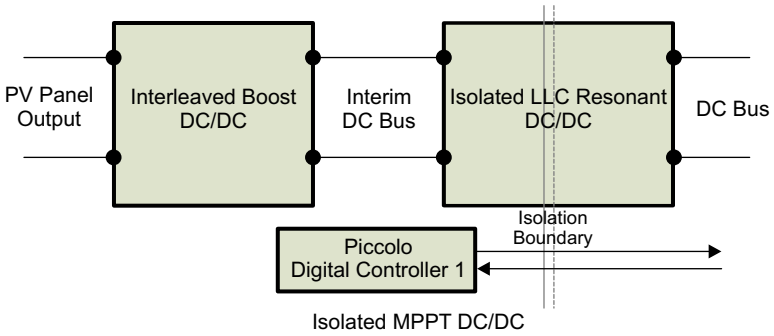


图 **1.** 隔离式 **DC-DC** 转换器方框图

此 DC-DC 转换器从 PV 电池板汲取 dc 电流，这样，此电池板运行在其最大功率传输点上。 这要求将电池板输出，也就是 DC-DC 转换器输入保持在一个由 MPPT 算法决定的电平上。 这在 2ph 交错式升压转换器级内执行。 此隔离式 LLC 谐振转换器仅仅为 DC-DC 级提供高频隔离。

一个具有片载 PWM，ADC 和模拟比较器模块的 C2000 piccolo 微控制器能够执行这样一个 MPPT DC-DC 系统的完全数字控制。

1.1 DC-DC 级实施

图 2图示了一个基于 C2000 的 MPPT DC-DC 转换器控制系统。 PV 电池板输出电压，Vpv，被施加到 2ph 交错式升压级。

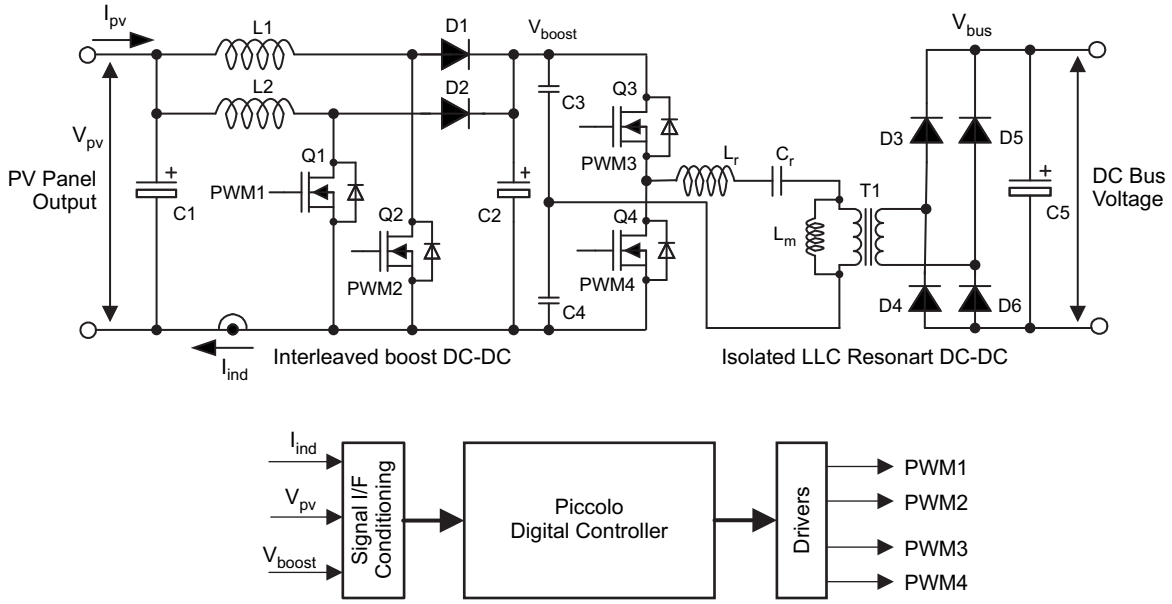


图 2. 使用 C2000 微控制器的 MPPT DC-DC 转换器控制

电感器 L1，MOSFET Q1 和二极管 D1 一起组成升压级中的一个，而 L2，Q2 和 D2 组成了另外一个升压级。 升压转换器输出上的一个电容器 C2 作为一个储能器并为谐振 LLC 级提供升压电压。

H 桥 LLC 谐振级包含 MOSFET Q3~Q4，输入电容器 C3~C4，谐振电感器 Lr，谐振电容器 Cr，变压器 T1，输出整流器 D3~D6 和输出电容器 C5。 这个级有一个为 1 的电压比，并提供初级侧与次级侧间的隔离。

图 1表示完全控制这个使用 C2000 微控制器 (MCU) 的 DC-DC 转换器所需的全部接口信号。 此 MCU 使用 3 个反馈信号和 4 个 PWM 输出来控制硬件。 被感测并被反馈回 MCU 的信号包括电池板输出电压 (Vpv) 和升压输出电压 (Vboost) 和总升压电感器电流 (I_{ind})。 这些感测到的信号被用来执行针对 DC-DC 升压级的电压和电流控制环路。 交错式升压 DC/DC 拓扑结构被选中来将易变的 DC 输出升压为一个固定的 DC 总线电压。 使用这个拓扑结构的主要原因是宽泛的输入电压变化。 针对电源开关 Q1 和 Q2 的 PWM 信号被相移 180 度。 这有助于减少 PV 电池板电流内的纹波。

此 LLC 级运行在开环路上，它的 PWM 频率被设定为与谐振频率一样。 Picclo 控制器与 LLC 级的初级侧共用接地，并且没有 LLC 次级输出端子到控制器的隔离反馈。 因此，LLC 运行在开环路上，所以，有必要保持一个值为 1 的电压转换因数。实现方法为：

- 使 LLC PWM 频率与谐振频率一样。
- 在 LLC 输出上保持一个大约为 10W 的最小负载。

图 3显示了 DC-DC 交错式升压转换器控制环路。 这使用电流模式控制。 然而，此目的是为了控制 PV 电池板输出 (Vpv)，此输出是到 DC-DC 级的输入。 这使得 PV 电池板（阵列）可始终运行在它的最大功率点上。 通过调整电源开关 Q1 和 Q2 的占空比，可调节输入电流。 通过调整输入电流来调节输入电压。 下个部分内描述的最大功率点跟踪算法负责确定 PV 电池板电压的设定点 (Vpv_ref)。

请注意，与输出电压控制中使用的传统反馈相比，输入电压控制环路的工作方式完全不同。 在这个控制机制下，当 PV 电池板电压 (Vpv) 有可能上升，高于 MPPT 算法设定的基准电池板电压 (Vpv_ref) 时，此控制环路增加电池板电流指令（针对内部电流环路 Iind_ref 的基准电流），从而将电路板电压控制在基准电平上 (Vpv_ref)。 当电池板电压有可能下降，低于此基准时，控制环路减少电池板电流指令，以将电池板电压复原至它的基准电平。

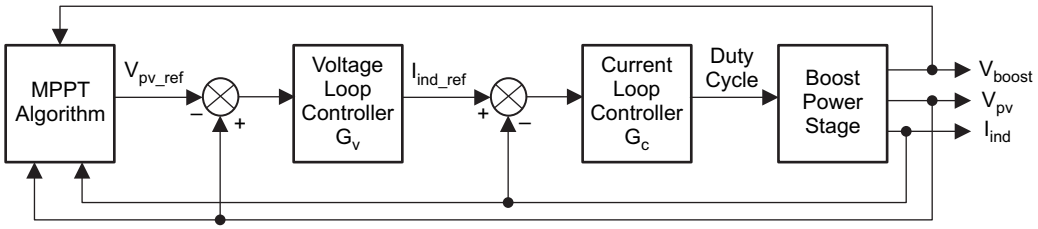


图 3. MPPT DC-DC 转换器控制环路

从 ADC 通道的其中一个通道感测到的电池板电压 Vpv 与 MPPT 算法设定的基准电压 Vpv_ref。 然后，得出的误差信号 Ev 被输入电压环路控制器 Gv，此控制器将电池板电压调节在基准电平上。 电压控制器 Gv 的形式为一个 2 极 2 零 (2P2Z) 补偿器。 Gv 的输出是针对内部电感器电流环路的基准电流指令。 电感器电流的平均值是电池板电流 Ipv。 因此，通过控制电感器电流的平均值，此电流控制器 Gc 从根本上控制电池板电流。

这个针对电流控制环路的基准电流指令 Iind_ref 与另外一个 ADC 通道感测到的反馈电感器电流 Iind 相比较。 然后，得出的电流误差信号被输入电流环路控制器 Gc，此控制器生成针对升压开关 Q1 和 Q2 的升压转换器 PWM 占空比指令。

除了执行电压和电流环路控制器，C2000 MCU 还监控针对过压保护的升压输出电压。 监控升压电压的 ADC 通道具有一个支持用户可编程阀值的内部模拟比较器。 这个针对比较器的阀值由一个内部 10 位 DAC 设定。 只要 DC 总线电压达到一个与用户可编程比较器阀值相对应的上限值，此比较器启动一个针对升压 PWM 信号的逐脉冲占空比限值。 它限制了升压电感器电流，并因此将升压总线电压限制在其想要的上限值。

C2000 MCU 还生成了 2 个 PWM 输出来驱动此隔离式 LLC 级。 这个级运行在一个开环方式下，此方式具有单位电压转换比（电压增益）。 这意味着此升压电压与 LLC 输出电压 Vbus 几乎相等。 然而，这要求 Vbus 上大约 10W 的小型最小负载（400V 时为 16kΩ）。 当 Vbus 上无负载且升压输出电压被设定为 400V 时，LLC 级增益必须大于 1，从而导致 LLC 输出上的高压 (Vbus)。 您必须通过始终在 LLC 输出上 (Vbus) 保持一个大约 16kΩ 的最小负载电容器来防止这一情况的发生。

所有与 DC-DC 控制环路相关的时间关键函数在一个快速采样环路内执行，此环路由 C2000 微控制器高速 CPU，中断，片载 12 位 ADC 模块和高频 PWM 模块启用。 此软件算法的详细说明在以下部分内提供。

1.2 DC-DC 电气技术规范

下面列出了 C2000 MPPT DC-DC EVM 的主要亮点

- 电池板电压：200V（最小值）至 300V（最大值）
- 400Vdc 输出
- 500W 输出功率
- 满负载效率大于 94%

2 软件概述

2.1 软件控制流程

此 Code Composer Studio™ 项目针对 C2000 MPPT DC-DC，它主要使用 “Cbackground/ ASM-ISR” 架构。 主快速 ISR (50kHz) 运行在汇编环境中。 然而，一个较慢的 ISR (10kHz) 也运行在 C 语言环境中。 这个慢速 ISR 可由快速 ISR 中断。 此外，一个第三 ISR 在 C 语言环境中以更慢频率运行以根据与 DC-AC 逆变器级通信来执行本地互连网络 (LIN)。 LIN 中断的频率由逆变器设定为 50Hz。

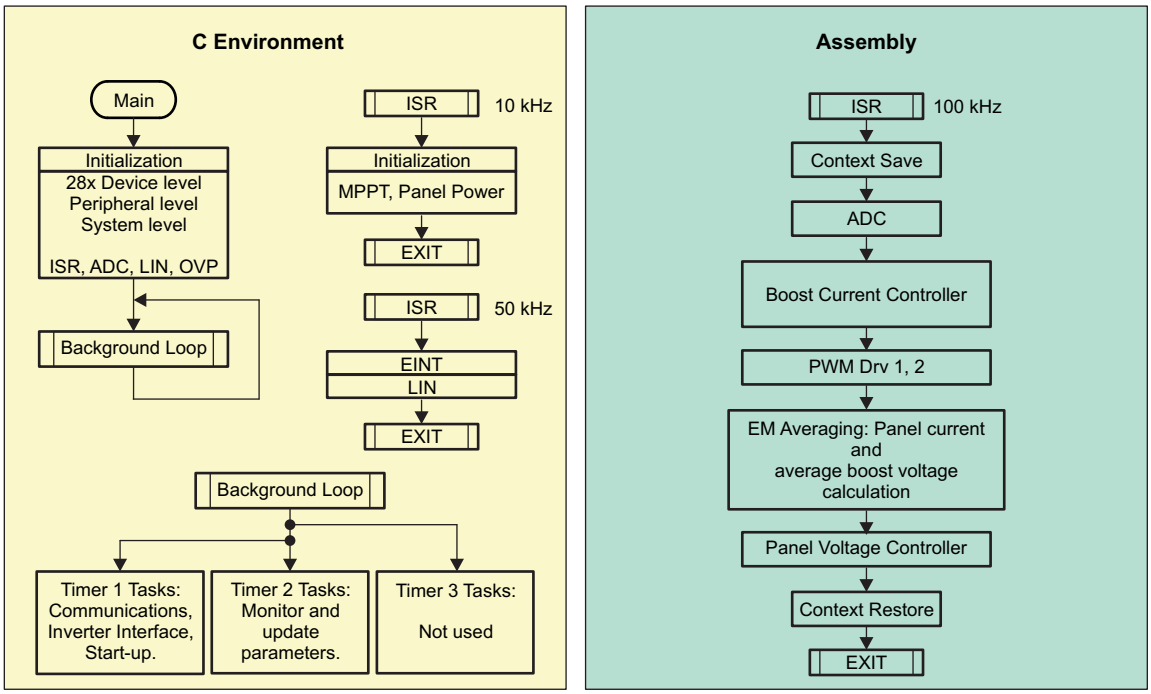


图 **4. MPPT DC-DC** 软件流程图

此项目使用 C 语言代码作为针对应用的主要支持程序并负责所有系统管理任务、决策制定、智能和主机交互。 此汇编代码被严格限制为快速中断处理例程 (ISR)，此例程运行全部关键控制代码。 通常情况下，这包括读取 ADC 值，控制计算和 PWM 更新。 C 语言环境中的较慢 ISR 执行 MPPT 算法来计算基准 PV 电池板电压并与逆变器级建立 LIN 通信。图 4图示了针对这个项目的普通软件流程图。

本项目中使用的关键框架 C 语言文件为：

- *HV_Solar_DC-DC-Main.c*：这个文件被用来初始化、运行和管理此应用。
- *HV_Solar_DC-DC -DevInit_F2803x.c*：MPPT DCDC EVM 中使用的 controlCard (2803x)，这些文件中的一个将位于 Code Composer Studio 项目中。 这个文件负责 F280xx 器件的一次初始化和配置，并且包括诸如设置时钟，锁相环 (PLL)，通用输入输出 (GPIO) 等函数。

快速 ISR 包含一个单个文件：

- *HV_Solar_DC-DC-DPL-ISR.asm*：这个文件包含所有时间关键“控制类型”代码。 这个文件有一个初始化部分（一次执行）和一个运行时间部分，此部分以用来触发它的 PWM 时基 (100kHz) 的一半速率执行。

  运行在 10kHz 上的慢速 ISR 包含两个文件： 用户选择这两个文件中的一个来执行 MPPT 算法。

- *Mppt_incc.h*：这个文件包含使用递增电感方法来计算用于最大功率点跟踪的电池板电压的代码。 这个文件有一个初始化部分（一次执行）和一个运行时间部分，此部分的执行速率为 10kHz。
- *Mppt_pno.h*：这个文件包含使用 扰动观察方法来计算用于最大功率点跟踪的电池板电压的代码。 这个文件有一个初始化部分（一次执行）和一个运行时间部分，此部分的执行速率为 10kHz。

运行在 50Hz 上的第二慢速 ISR 包含一个文件。

- *SolarHv_DCDC_Lin.C*：这个文件包含用于建立与逆变器级 LIN 通信的代码。

从快速 ISR 架构“调用”电源库函数（模块）。

这些电源库模块也许具有 C 语言和汇编元素。 在这个项目中，使用了 5 个库模块。 C 语言和相对应的汇编模块名称为：

表 1. 库模块

C 配置函数	**ASM** 初始化宏	**ASM** 运行时间宏
PWM_1ch_UpDwnCnt_Cnf.c	PWMDRV_1ch_UpDwnCnt_INIT n	PWMDRV_1ch_UpDwnCnt n
ADC_SOC_Cnf.c	ADCDRV_1ch_INIT m,n,p,q	ADCDRV_1ch m,n,p,q
PWM_CompPairDB_Cnf.C		
	MATH_EMAVG_INIT n	MATH_EMAVG n
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

此汇编模块也可如图 5中那样进行图形化表示。

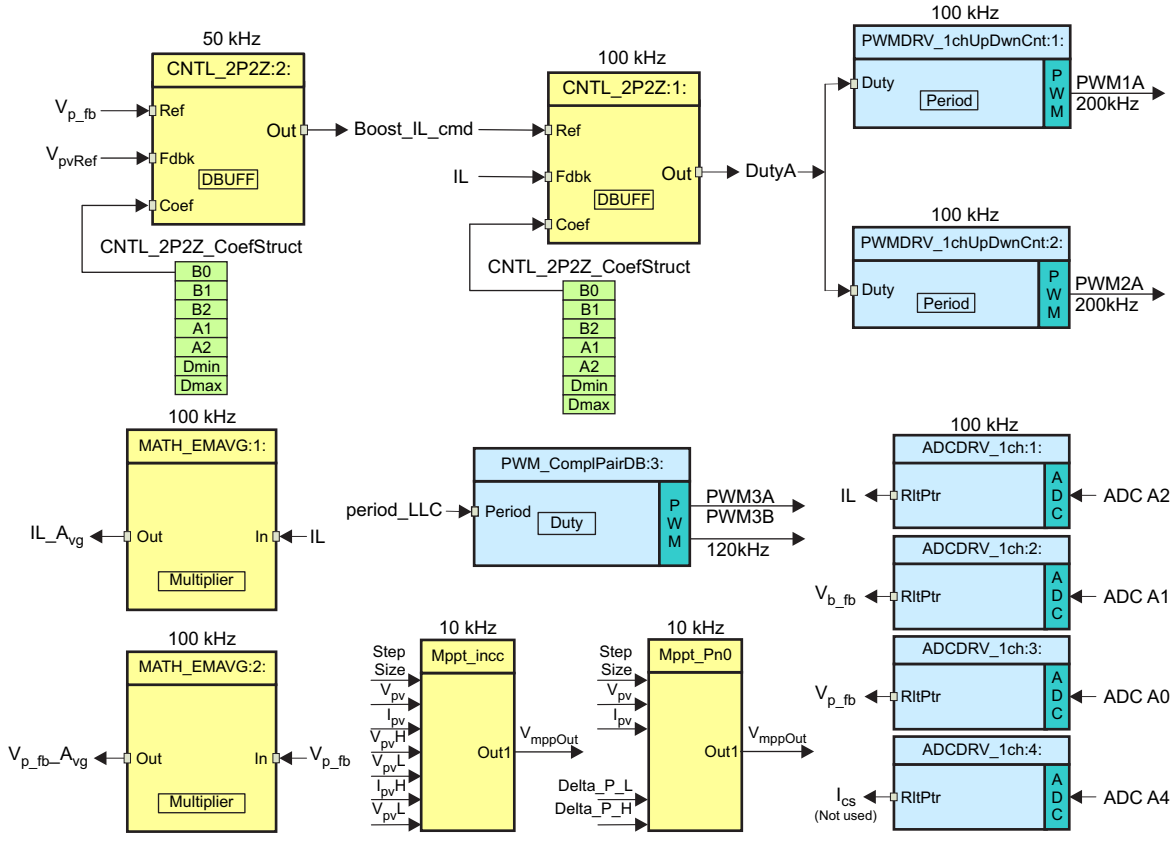


图 5. 软件区块

请注意针对图 5中模块的颜色编码。 ‘深蓝色’的区块代表 C2000 控制器上的片载硬件模块。 ‘蓝色’区块是这些模块的驱动程序。 ‘黄色’区块是不同信号上执行的计算部分。 用于电压和电流环路的控制器的形式为一个 2 极 2 零比较器。 然而，可以为其它形式，诸如，PI，PID，3 极 3 零或任何适用于此应用的其它控制器。 这样一个模块库结构使得图形化和理解图 6中显示的整个系统软件流程变得更加方便。 它还可实现多种功能性的简单使用、添加/删除。 通过执行一个递增构建方法可在这个项目中充分证明这一事实。 这在下一部分进行了更加详细的讨论。

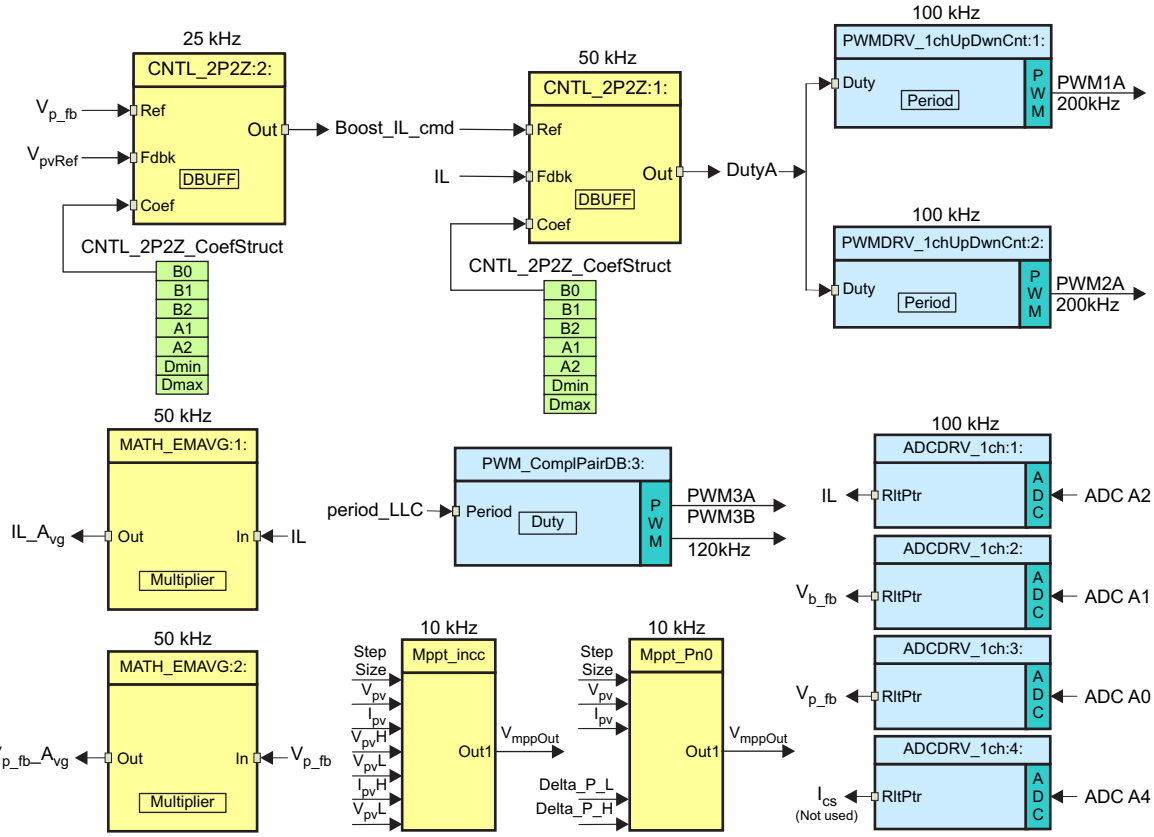


图 6. 软件控制流程

如1.1 节中所述，MPPT DC-DC 系统由两个反馈环路控制。 此外部电压环路将电池板电压保持在 MPPT 算法计算得出的电平上，而一个较快速电流环路控制平均升压电感器电流。图 6还给出了软件模块的执行速率。 例如，电流控制器的执行速率为 50kHz（PWM 开关频率的一半），而电压控制器的执行速率为 25kHz。

2.2 递增构建

此项目被分成 3 个递增构建。 这个方法为用户提供一个逐步方法来熟悉此软件，并了解它是如何与 MPPT DC-DC 硬件进行交互的。 这个方法还简化了调试任务以及电路板测试。

表 2中显示了构建选项。 将一个*HV_Solar_DC-DC-Settings.h*文件中的特定构建选项集 INCR_BUILD 选入表 2中所示的相应构建选择中。 一旦选择了构建选项，通过选择重建全部编译器选项来编译整个项目。 下个部分提供了与如何运行每个构建选项的更多细节。

表 2. 针对 MPPT DC-DC 的递增构建选项

递增构建选项	
INCR_BUILD = 1	针对升压和 LLC 操作以及 ADC 反馈的开环路检查（检查感测电路）
INCR_BUILD = 2	升压的开电压环路和闭合电流环路控制
INCR_BUILD = 3	具有 MPPT 的升压的闭合电压和电流环路控制

3 运行递增构建的步骤 - 全部

所有与这个 C2x 控制的 MPPT DC-DC 系统相关的软件文件（主源文件，ISR 汇编文件和针对 C 语言框架的项目文件）位于目录www.ti.com/controlsuite - (\development_kits\MPPT DC-DC_v1.0\MPPT DC-DC) 中。 与这个软件在一起的项目针对 Code Composer Studio v4。

CAUTION

在电路板上会有高压出现。 它应该只在实验室环境中由经验丰富的电源专业人员处理。 为了安全评估这个电路板，应该使用一个具有适当功率额定值的 PV 电池板仿真器来为此部件供电。 在电源被施加到电路板上之前，必须将一个电压计和一个适当的阻性或电力负载连接至输出。 当 PV 电源被关闭时，将快速将总线电容器放电。 在电路板上未执行输出过流保护，所以，您应当采取适当的措施来防止任何输出短路情况的发生。

以下步骤被用来构建和运行包括在 DC-DC 软件内的示例。

3.1 构建 1：具有 ADC 测量的开环路升压

3.1.1 目的

这个构建的目的在于：

- 评估 MPPT DC-DC PWM 和 ADC 软件驱动器模块
- 验证 MOSFET 栅极驱动电路、电压和电流感测电路
- 熟悉 Code Composer Studio 的操作

使用这个构建，此系统运行在开环模式下，所以被测得的 ADC 值只用于电路验证和仪表应用。 下面解释了构建和运行一个 Code Composer Studio 项目所需的步骤。

3.1.2 概述

已经配置了 Build1 中的软件，这样您就能够通过查看示波器上的相关波形并且观察占空比变化对 DC-DC 输出电压的影响来快速评估 PWM 驱动器模块。 用户能够从 Code Composer Studio 观察窗口中调整 PWM 占空比。 用户还可以通过查看观察窗口中的 ADC 采样数据来评估 ADC 驱动器模块。

PWM 和 ADC 驱动器宏例示在 _DPL_ISR 内部执行。图 7显示了这个构建中所用的软件区块。 针对 2 个 BOOST 开关的 2 个 PWM 信号从 ePWM 模块 1 和 2 获得。ePWM1A 驱动 BOOST 开关中的一个，而 ePWM2A 驱动另外一个。 针对 2 个 LLC 级的 2 个 PWM 信号从 ePWM 模块 3 获得。ePWM3A 驱动上部（高侧）LLC，而ePWM3B 驱动低侧开关。

被感测和反馈回 MCU 的数量包括：

- 电池板电压 (Vp_fb)
- 组合 BOOST 电感器电流 (IL)
- 升压 DC 总线电压 (Vb_fb)

这些量由 ADC 驱动器模块读取并在图 7中显示。 这个应用中未使用针对 ADC 驱动器宏的第四个通道。 ADC 驱动器模块将 12 位 ADC 结果转换为一个 32 位 Q24 值。

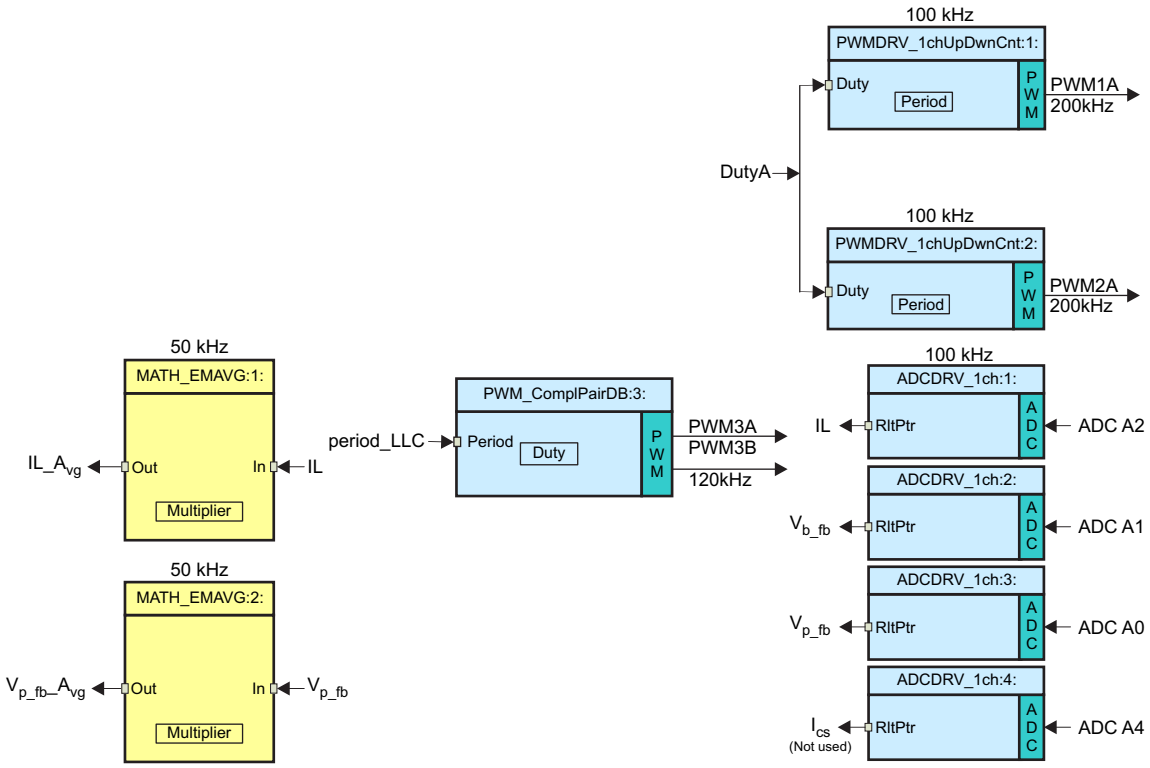


图 7. 构建 1 软件区块

升压 PWM 信号的生成频率为 100kHz（即，一个 10µs 周期）。 当控制器运行频率为 60MHz 时，ePWM1 的时基计数器的一个计数的相应值为 16.6667ns。 这意味着一个 10µs 的 PWM 周期将等于时基计数器（TBCNT1，TBCNT2）的 600 次计数。 如图 8中所示，ePWM1 和 ePWM2 模块被配置成运行在上-下数模式。 这意味着一个 300 的时基周期值（周期寄存器值）将提供一个总数为 600 次计数的 PWM 周期值 (10µs)。

LLC PWM 信号的生成频率为 120kHz（即，一个 8.33µs 的周期）。 当控制器运行频率为 60MHz 时，ePWM3 的时基计数器的一个计数的相应值为 16.6667ns。 这意味着一个 8.33µs 的 PWM 周期将等于时基计数器 (TBCNT3) 的 500 次计数。 ePWM3 模块被配置成运行在上-下计数模式。 这意味着一个 250 的时基周期值（周期寄存器值）将提供一个总数为 500 次计数的 PWM 周期值 (8.33µs)。

由于已采样的值表示 CCM（持续传导模式）条件下的平均电感器电流，BOOST 电感器电流在 PWM1A ON 脉冲的中点采样。 在 DCM（不连续传导模式）条件下，这个采样电流表示平均电感电流的一部分。

另外两个电压信号转换也在此时启动。 图 8中显示了这一操作。 C2000 器件上 ADC 和 PWM 模块的灵活性可实现精准且灵活的 ADC 转换启动。 在这个情况下，ePWM1 被用作一个时基来在 TBCNT1 达到零值时，生成一个转换启动 (SOC) 触发。 为了确保 ADC 结果的完整性，在这个点上执行一个假 ADC 转换。

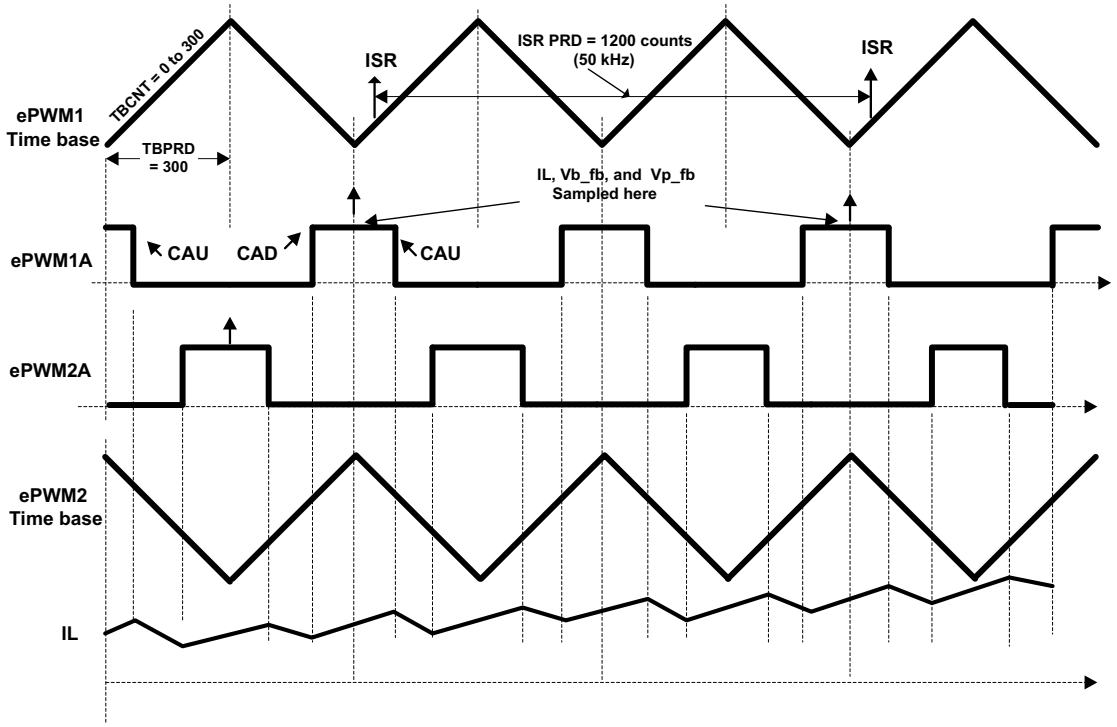


图 **8. PWM** 生成和 **ADC** 采样

在一个 CAU 时间发生时（TBCNT1 = CMPA 并且上计数），ePWM1A 输出被复位，而一个 CAD 事件发生时（TBCNT1 = CMPA 并且下计数），ePWM1A 输出被置位。ePWM2A 的配置方式类似，相对于 ePWM1A 有 180 度相移。

CMPA 值得自输入“BOOSTDuty”（Q24 变量）命令。

ADC 模块被配置成使用 ePWM1 的 SOCA，这样，SOCA 在 TBCNT1 = ZERO 事件发生时被触发。 使用这个 SOCA 触发来完成所有转换。 通过从被标记为 _DPL_ISR 的 50Hz ISR 内执行 ADC 驱动器模块，来在 ISR 中读取这 3 个 ADC 结果。

在上计数的 CMPB 匹配事件发生时，汇编中的 ISR (_DPL_ISR) 由 EPWM1 触发。 CMPB 被设定为 80，这样 ISR 只在 ADC 转换完成后被触发。 这是 PWMDRV_1ch_UpDwnCnt 宏被执行，以及PWM 比较阴影寄存器被更新的地方。 这些在下一次 TBCNT=ZERO 事件发生时被载入到活动寄存器中。 请注意，ISR 触发频率是图 8中所示的 PWM 开关频率的一半。

3.1.3 保护

针对这个 MPPT DC-DC EVM，在软件中执行一个过压保护机制。 这个 OVP 只应用于升压输出，不用于 LLC 级输出。 由于 Piccolo 控制器位于隔离的初级侧上，它并不知道隔离式 LLC 输出。 因此，为了使得开环 LLC 输出符合它的输入电压，升压级的输出电压，用户必须在 LLC 输出上跨接一个大约 10W 的最小负载。 LLC 输出上跨接的最小负载帮助将 LLC 级电压转换因子保持为 1。使用这个方法，最大 LLC 输出也将受到最大升压输出的限制，此最大升压输出反过来受到 OVP 机制的保护。

来自 ADC 输出的感测升压级 DC 总线输出电压与用户设定的过压保护阀值相比较。 缺省 OV 阀值设定点为 404V。这个阀值参数在文件*HV_SOLAR_DC_DCMain.h*内设定。 这通过以下的片载 DAC 基准电压初始化来完成。

Comp3Regs.DACVAL.bit.DACVAL = 808;

由于 10 位 DAC 满量程值代表一个 511V 的最大电压，808 的初始值代表 404V 的 OV 阀值。

在 OV 情况下，PWM 输出是使用 TZ（触发区）寄存器的占空比受限逐脉冲。 C2000 器件上触发机制的灵活性提供根据不同触发事件采取不同操作的可能性。

3.2 过程

3.2.1 启动 Code Composer Studio 并打开一个项目

下面的步骤被用来快速执行这个构建：

1. 为了实现仿真，将 USB 连接器接至 Piccolo 控制器电路板。
2. 在 JP1 上为 12V 偏置电源加电。 缺省情况下，Piccolo 控制卡跳线（请见 Piccolo controlCard 文档）被配置成让器件从闪存引导。 改变这些跳线设置，来允许 Code Composer Studio 控制下，从 RAM 执行代码。
3. 启动 Code Composer Studio。 一个项目包含生成一个可执行输出文件 (.out) 所需的全部文件和构建选项，此文件能够在 Code Composer Studio 内，MCU 硬件上运行。
4. 在菜单栏上单击 Project → Import Existing CCS/CCE Eclipse Project（项目 → 导入现有的 CCS/CCE Eclipse 项目）。 在选择根目录下 → 导航至并选择目录：www.ti.com/controlsuite - (\development_kits\MPPT DC-DC_v1.0\MPPT DC-DC)。 请确保选中项目标签页下的 MPPT DC-DC。 单击完成。

   这个项目调用构建项目所需的全部工具（编译器、汇编程序和连接程序）。
5. 在左侧的项目窗口中，单击项目左侧的加号符号 (+)。 您的项目窗口与图 9中显示的内容类似。

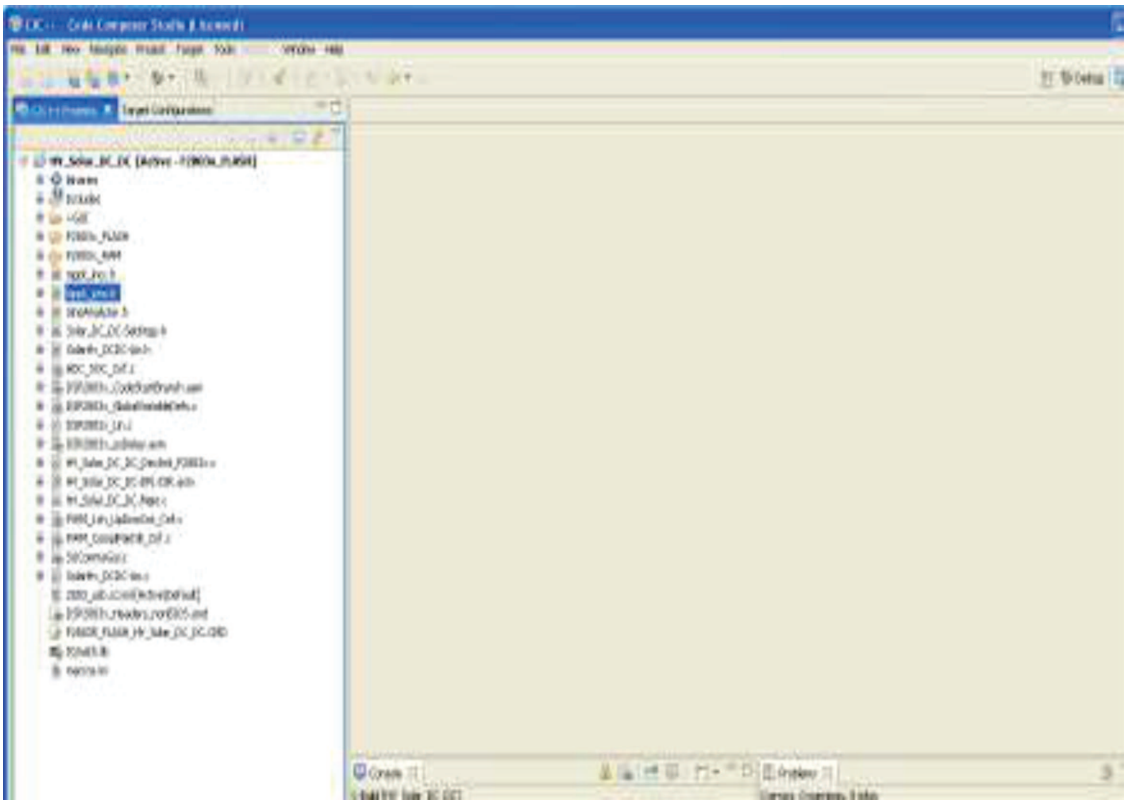


图 **9. Code Composer Studio** 项目窗口

3.2.2 器件初始化，主和 ISR 文件

注: *不要*对源文件做出任何改动 - *检查即可*。

1. 通过双击项目窗口中的文件名可打开并检查*SOLAR_DC_DC-DevInit_F2803x.c*。 请注意，系统时钟、外设时钟预分频和外设时钟使能已经被设置。 请注意共用 GPIO 引脚已经被配置。
2. 打开并检查*SOLAR_DC_DC -Main.c*. 请注意对*DeviceInit()*函数的调用和其它变量初始化。 此外，请注意针对不同递增构建选项，ISR 初始化以及 (;;) 环路所用背景的代码。
3. 确定并检查以下专门针对构建 1 的初始化代码下，主文件内的代码。这是控制流程内*PWMDRV_1ch_UpDwnCnt*和*ADCDRV_1CH*区块被连接的地方。

```
//-------------------------------------------------------------------------------------------------
#if (INCR_BUILD == 1)   // Open Loop Debug for Boost Stage; LLC stage always runs under open loop
//-------------------------------------------------------------------------------------------------
    // Lib Module connection to "nets"
    //-----------------------------------------
    // Connect the PWM Driver input to an input variable, Open Loop System
    ADCDRV_1ch_Rlt1 = &IL_raw;//Raw ADC data which seems to have some nonzero offset
    ADCDRV_1ch_Rlt2 = &Vb_fb;
    ADCDRV_1ch_Rlt3 = &Vp_fb;
    ADCDRV_1ch_Rlt4 = &Ics;

    // Math_avg block connections - Instance 1
    //MATH_EMAVG_In1=&IL_raw;
    MATH_EMAVG_In1=&IL;//Input instantaneous IL after offset correction (correction done is ISR)
    MATH_EMAVG_Out1=&IL_avg;//Output Avg IL after offset correction
    MATH_EMAVG_Multiplier1=_IQ30(0.030);

    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vb_fb;
    MATH_EMAVG_Out2=&Vb_fb_Avg;
    MATH_EMAVG_Multiplier2=_IQ30(0.00025);

    PWMDRV_1ch_UpDwnCnt_Duty1 = &DutyA;
    PWMDRV_1ch_UpDwnCnt_Duty2 = &DutyA;
    PWMDRV_1ch_UpDwnCnt_Duty4 = &Duty4A;

    // Initialize the net variables
    DutyA =_IQ24(0.1);//Variable initialized for open loop test of the DC-DC Boost Stage
    DutyLLC =_IQ24(0.5);
    Duty4A =_IQ24(0.0);
    IL_avg = _IQ24(0.0);
    IL = _IQ24(0.0);
    IL_raw = _IQ24(0.0);
    Vb_fb_Avg = _IQ24(0.0);

#endif // (INCR_BUILD == 1),
```

4. 确定并检查以下初始化代码下，主文件内的代码。 这是*PWMDRV_1ch_UpDwnCnt*区块被配置和初始化的地方。 这对于所有递增构建通用。 这个 PWM 驱动器模块输入 600 的总 PWM 周期值，并在内部计算出 300 的周期寄存器值。

```
//=============================================================================
//  INCREMENTAL BUILD OPTIONS - NOTE: selected via (Solar_DC_DC-Settings.h
//=============================================================================
// ---------------------------------- USER ----------------------------------------

#define period 600  //600 cycles -> 100KHz @60MHz CPU, PWM period for 2Ph Interleaved Boost stage
#define phase 300   //Phase shift for slave PWM in 2Ph IL Boost stage
#define period_LLC 500 //545=>110kHz,,460=>130kHz, 500 cycles -> LLC freq 120KHz @60MHz CPU
//#define period_LLC_NO_LOAD 460 //460=>130kHz, -> LLC freq 130KHz @60MHz CPU
//#define period_instr_pwm 120  //120 => 500k @60MHz CPU, period for PWM channels used for instrumentation

    // Configure PWM1 for 100Khz switching Frequency
    PWM_1ch_UpDwnCnt_CNF(1, period, 1, 0);
    // Configure PWM2 for 100Khz switching Frequency
    PWM_1ch_UpDwnCnt_CNF(2, period, 0, 300);


    PWM_ComplPairDB_CNF(3, period_LLC, 1, 0);
    (*ePWM[3]).CMPA.half.CMPA = period_LLC/2;

    //PWM_ComplPairDB_UpdateDB(3,25,25);
    //PWM_ComplPairDB_UpdateDB(3,35,35);//Trying this in Rev2 board for higher efficiency
    PWM_ComplPairDB_UpdateDB(3,dbred,dbred);
```

5. 确定并检查以下初始化代码下，主文件内的代码。 这是 ADCDRV_1CH 区块被配置和初始化的地方。 这对于所有递增构建也是通用的。

```
#define     ILR     AdcResult.ADCRESULT1        //Q12
#define     Vb_fbR  AdcResult.ADCRESULT2        //Q12
#define     Vp_fbR  AdcResult.ADCRESULT3        //Q12
#define     IcsR    AdcResult.ADCRESULT4        //Q12

          // ADC Channel Selection for C2000EVM
    ChSel[0] = 2;       // Dummy read for first
    ChSel[1] = 2;       // A2 - ILb_fb, Boost inductor current
    ChSel[2] = 1;       // A1 - Vb_fb, Boost output volt
    ChSel[3] = 0;       // A0 - Vp_fb, Panel voltage
    ChSel[4] = 4;       // A4 - Ics_fb, LLC stage primary current

    // ADC Trigger Selection
    TrigSel[0] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
    TrigSel[1] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
    TrigSel[2] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
    TrigSel[3] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
    TrigSel[4] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA

    // Configure ADC
    ADC_SOC_CNF(ChSel, TrigSel, ACQPS, 17, 0);

    // Configure ePWMs to generate ADC SOC pulses
    EPwm1Regs.ETSEL.bit.SOCAEN = 1;                 // Enable ePWM1 SOCA pulse
    EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTR_ZERO;      // SOCA from ePWM1 Zero event
    EPwm1Regs.ETPS.bit.SOCAPRD = ET_1ST;            // Trigger ePWM1 SOCA on every event


    DPL_Init();
```

6. 打开并检查*SOLAR_DC_DC-DPL-ISR.asm*. 请注意构建 1 下的 _DPL_Init 和 _DPL_ISR 部分。这分别是 PWM 和 ADC 驱动器宏例示完成初始化和运行结束的地方。

3.2.3 构建和加载项目

1. 在*SOLAR_DC_DC-Settings.h*文件中将递增构建选项选为 1。

> **注:** 只要您改变了*SOLAR_DC_DCSettings.h*中的递增构建选项，请始终进行“重建全部”操作。

2. 单击 Project → "Rebuild All"（项目 → “重建全部”）按钮并观察在构建窗口中运行的工具。
3. 单击 Target → "Debug Active Project"（目标 → “调试激活项目”）。 如果还未选择，Code Composer Studio 将要求您打开一个新目标配置文件。 如果已经为这个连接创建了一个有效目标配置文件，您可前往步骤 1（在节 3.2.5中）。 在 New targe Configuration Window（新目标配置窗口）内，为您将操作的目标键入以 .ccxml 为后缀的文件名（例如：xds100-F28035.ccxml）。 选中 "Use shared location"（使用共用位置） 并单击 Finish。
4. 在打开的 .ccxml 文件中，将 Connection（连接）选为 "Texas Instruments XDS100v2 USB Emulator"（德州仪器 (TI) XDS100v2 USB 仿真器）。 向下滚动并选择 "TMS320F28035" 器件。 单击

Save（保存）。

5. 单击 Target → "Debug Active Project"。 将项目配置选为 F2803x_FLASH。 此程序将被载入到闪存内。 您现在应该处于 Main() 的起始位置。

3.2.4 调试环境窗口

在调试代码时观察本地和全局变量是标准调试做法。 在 Code Composer Studio 内有多种不同的方法来进行这个操作，诸如存储器视图和观察视图。 如果调试环境启动时观察视图没有打开，那么打开一个新的观察视图并通过下面给出的步骤为其添加不同参数。

1. 单击菜单栏上的 View → Watch（视图 → 观察）。
2. 单击顶部观察视图上的 “Watch (1)" 标签页。 您可将任一变量添加至观察视图。
3. 在 "Name" 栏内的空白方框内输入您想观察的变量符号名并且按下键盘上的回车键。 请确保按要求修改了 "Format"（格式）。

3.2.5 使用实时仿真

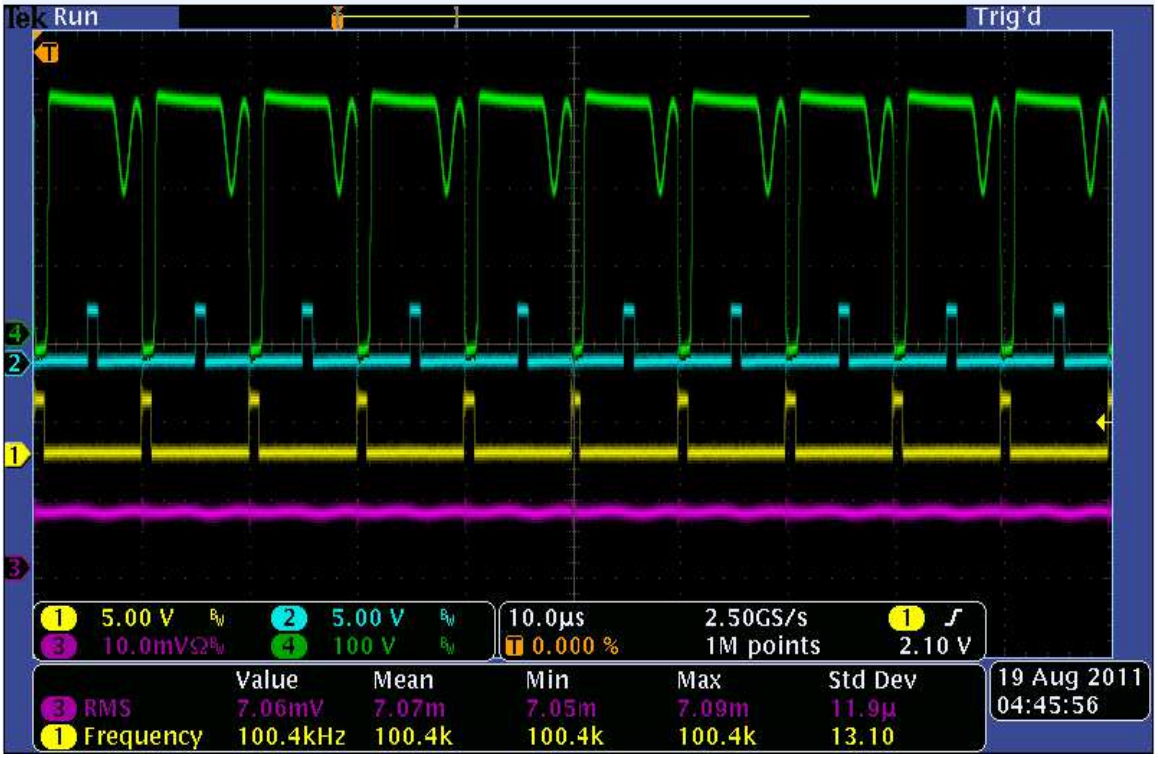
实时仿真是一种特殊的仿真特性，它可以让 Code Composer Studio 中的窗口在 MCU 运行的时候以高达 10Hz 的速率刷新。 这不但可实现图形和观察视图更新，而且使您能够改变观察和存储器窗口内的值，并且查看这些变化在系统中的影响。

1. 通过将您的鼠标悬停在水平工具条的按钮上并单击  （*Enable Silicon Real-Time Mode*（启用芯片实时模式）（当暂停时处理关键中断，可在运行时访问调试程序）按钮来启用实时模式。
2. *也许*会出现一个消息框。 如果是这样的话，选择 YES 来启用调试事件。 这将把状态寄存器 1 (ST1) 的位 1（DGBM 位）置位为“0”。 DGBM 是调试使能屏蔽位。 当 DGBM 位被置位为“0”时，为了更新调试器窗口，存储器和寄存器值可被传递到主机处理器。
3. 对于观察视图，请在  上单击。

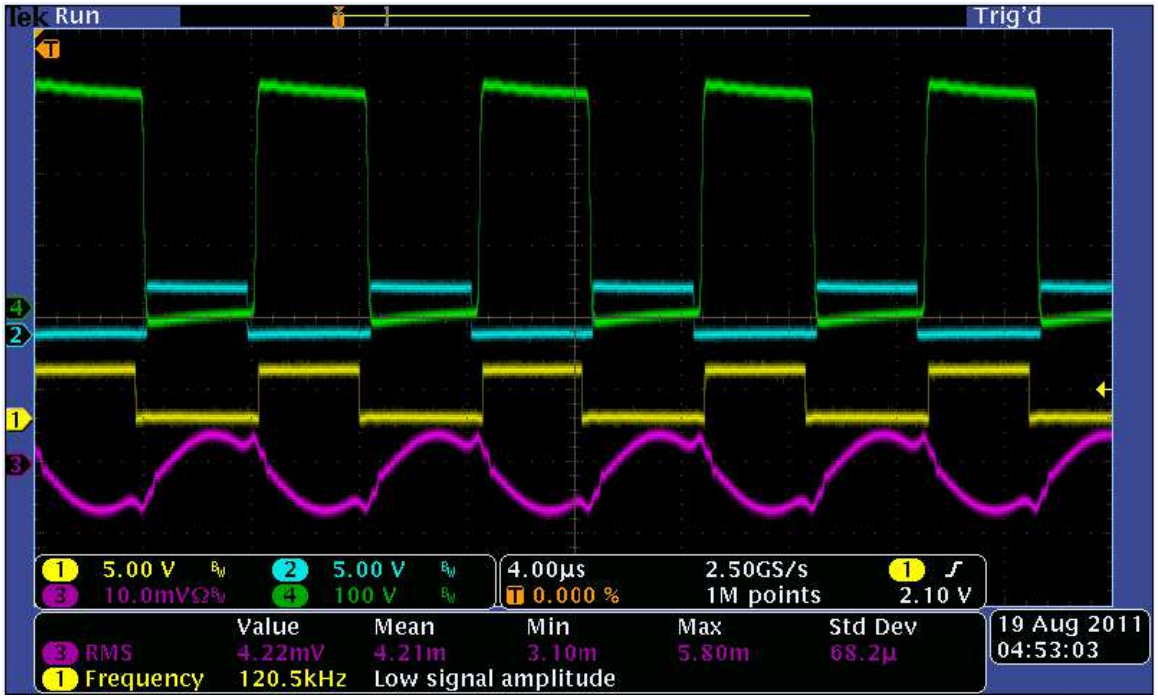
3.2.6 运行代码

1. 使用 <F8> 键来运行代码，或者使用工具条上的 RUN（运行）按钮，或者使用菜单栏上的 Target → Run（目标 → 运行）。
2. 在观察视图中，添加变量 DutyA 并将其设定为 0.1 (等于采样 Q24 格式表示法中的 1677721)。 这个变量为升压转换器设定占空比。
3. 将一个阻性负载施加到 DC-DC EVM 输出端子上 (10-100W)。
4. 使用一个高压隔离式 DC 电源为 DC-DC EVM 供电。 测量并验证与施加的输入电压和占空比相对应的升压 DC 总线电压。
5. 使用 Duty A 来缓慢改变观察窗口的占空比。 升压转换器输出电压应该发生相应地变化，而这反过来将改变 EVM 输出。

   仔细观察输出电压，不应该让这个值超过电路板的最大额定电压。
6. 在观察视图中添加其它诸如 Vb_fb，Vp_fb 的变量，并验证不同的 ADC 结果。
7. 下面示波器捕捉显示了 2 个 PWM 输出（Ch1 和 Ch2），DC 源输入电流 (Ch3) 以及升压 MOSFET 漏源电压 (Ch4)，此时输出 DC 总线负载为 1KΩ，输入 DC 电压为 250V，并且将占空比设定为大约 10%。 测得的 PMW 频率为 100kHz。



8. 下面的示波器捕捉显示了针对 LLC 级的 2 个 PWM 输出（Ch1 和 Ch2）。 在这个图表中，Ch4 代表 LLC 级初级开关节点电压，而 Ch3 为 LLC 初级电流，此时升压输出为 250V，升压输出（LLC 输入）为 300V，LLC 输出为 307V，LLC 输出负载为 1KΩ，而升压占空比被设定为大约 10%。



9. 尝试不同的占空比值，并观察相应的 ADC 结果。 以小步长增加占空比值。 始终小心观察输出电压，这个值不应超过电路板的处理能力。 诸如 PWM 栅极驱动信号、输入电压和电流以及输出电压等的不同波形也可使用一个示波器进行探测和验证。 探测这些高压信号时，请采取适当的安全措施。
10. 实时模式时，完全停止 MCU 是一个两步过程。 在 DC 输入关闭时，一直等到 DC 总线电容器完全放电。 首先，通过使用工具条上的 Halt（暂停）按钮，或者使用 Target → Halt（目标 → 暂停）来暂停处理器。 然后使 MCU 离开实时模式。 最后复位 MCU。
11. 您可以使 Code Composer Studio 继续运行，用于下一个应用，或者选择将其关闭。

3.3 构建 2 - 具有闭合电流环路的 MPPT DC-DC

3.3.1 目的

这个构建的目的在于验证 MPPT DC-DC 在闭合电流环路和开电压环路模式下的运行。 由于电压环路是开放的，在这个构建下没有 MPPT 操作。

3.3.2 概述

图 10显示了这个构建中所用的软件区块。 请注意，一个额外的软件区块与构建 1 图相比较 (图 7)。 这个区块在图 10中显示为*CNTL_2P2Z:1*。 它表示一个 2 极 2 零 (2p2z) 控制器，并用于电流控制环路。 根据控制环路的需求，可使用诸如一个 PI 或一个 3 极 3 零等其它控制区块。

如图 10中所示，电流环路区块的执行频率为 50Hz CNTL_2P2Z 是一个由无限脉冲响应 (IIR) 滤波器结构实现的二阶比较器。 这个函数与任何外设无关，因此就不需要 CNF 函数调用。

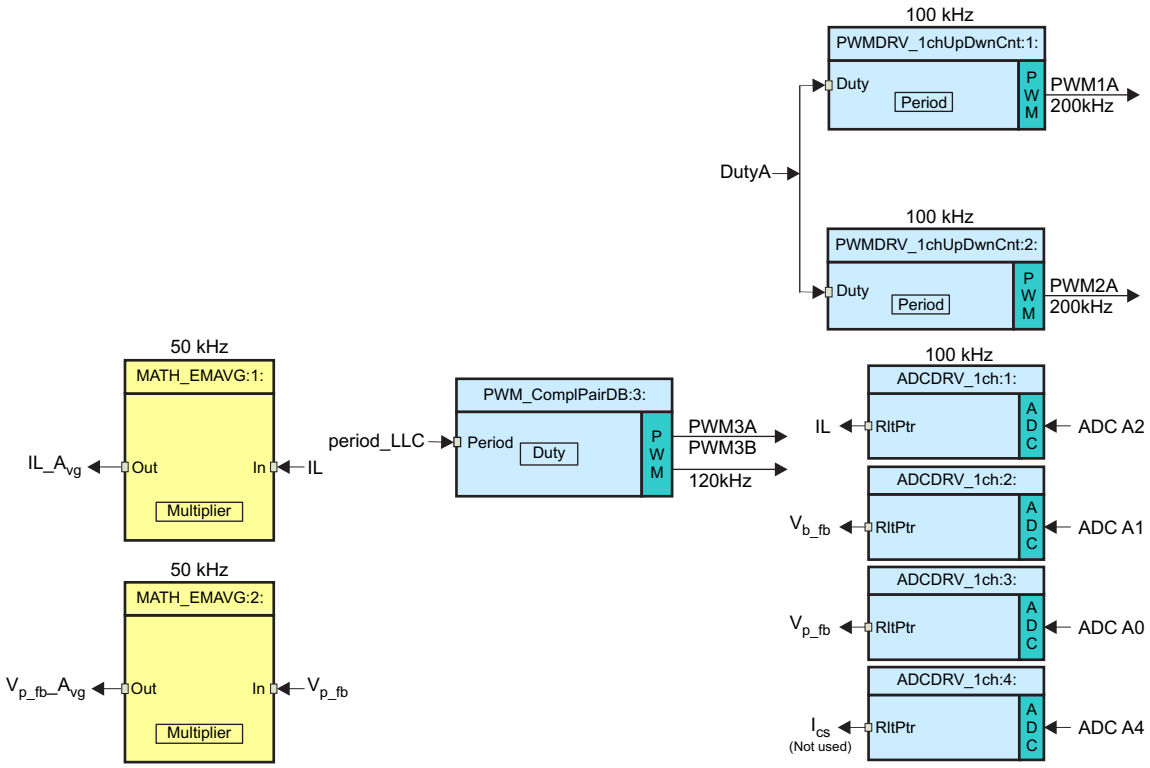


图 **10.** 构建 **2** 软件区块

这个 2p2z 控制器需要 5 个控制系数。 这些系数和控制器的钳位输出被存储为一个名为*CNTL_2P2Z_CoefStruct1*的结构的元素。 如果系统需要多个环路，CNTL_2P2Z 区块可被实例化多次。 每个实例可具有单独的系数集合。 针对电流环路的*CNTL_2P2Z*例示使用被存储为结构*CNTL_2P2Z_CoefStruct1*的元素的系数。 在这个方法中，*CNTL_2P2Z*的一个具有不同结构，*CNTL_2P2Z_CoefStruct2*的第二例示，可被用于电压环路控制，正如您将在下一个构建 3 部分中看到的那样。

可同构修改结构*CNTL_2P2Z_CoefStruct1*内 B0，B1，B2，A1 和 A2 值来直接更改控制器系数。 或者，2p2z 控制器可表示为 PID 形式，并且可通过改变 PID 系数来改变此系数。 下面显示了在 5 个控制器系数与 3 个 PID 增益间建立关联的等式。 对于电流环路，这些 P，I 和 D 系数被分别命名为：Pgain_I，Igain_I 和 Dgain_I。 对于构建 3 中使用的电压环路，这些系数被分别命名为：Pgain_V，Igain_V 和 Dgain_V。 这些系数采用 Q26 格式。

此补偿器区块 (*CNTL_2P2Z*) 有一个基准输入和一个反馈输入。 这个被标记为*Fdbk*的反馈输入来自 ADC。 这个被标记为*Ref*的基准输入通常来自电压环路控制器输出。 但是，在这个构建中，没有电压环路控制器，所以，在用户控制下，变量 Boost_IL_cmd 被用来改变基准电流。 针对*CNTL_2P2Z*的 z 域传输函数在以下给出：

$$\frac{U(Z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a1 z^{-1} + a2 z^{-2}} \tag{1}$$

PID 控制器的递归形式由公式 2给出：

$$u(k) = u(k-1) + b_{0e}(k) + b_{1e}(k-1) + b_{2e}(k-2) \tag{2}$$

其中，

$$b_0 = K_p + K_i + K_d$$
$$b_1 = K_p + K_i - 2K_d$$
$$b_2 = K_d \tag{3}$$

并且，这个 PID 的 z 域传输函数为：

$$\frac{U(Z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + z^{-1}} \tag{4}$$

将它与普通形式相比较，您可发现 PID 是 CNTL_2P2Z 控制的一个特殊情况，

其中，

$$a1 = -1 \quad \text{and} \quad a2 = 0 \tag{5}$$

显示在图 10中的指数移动平均 (MATH_EMAVG) 区块计算升压电感器电流和电池板电压的平均值。 平均电感器电流表示电池板电流，并被用来执行 MPPT 算法。

3.4 过程

3.4.1 构建和加载项目

按照与构建 1 中的步骤完全一样的步骤 1（来自节 3.2.1）至 5（来自节 3.2.2），(3.1 节)，除了那些 6中的步骤，来选择构建 2 选项，而非构建 1。然后按照下面所示完成步骤 6。

确定并检查主文件内初始化代码下面的专门用于构建 2 的代码。这是控制流程中与构建 2 相关的所有软件区块的连接位置。

```
//-----------------------------------------------------------------------------------------------------
#if (INCR_BUILD == 2)   // Closed Current Loop, Open Volt Loop boost config; LLC stage always runs under open loop
//-----------------------------------------------------------------------------------------------------
// Lib Module connection to "nets"
    //-----------------------------------------
    // Connect the PWM Driver input to an input variable, Open Loop System
    ADCDRV_1ch_Rlt1 = &IL_raw;//Raw ADC data which seems to have some non-zero offset
    ADCDRV_1ch_Rlt2 = &Vb_fb;
    ADCDRV_1ch_Rlt3 = &Vp_fb;
    ADCDRV_1ch_Rlt4 = &Ios;

    // Math_avg block connections - Instance 1
    //MATH_EMAVG_In1=&IL_raw;//*****Change to this for testing with rev 2 board
    MATH_EMAVG_In1=&IL;//Input instantaneous IL after offset correction (correction done in ISR)
    MATH_EMAVG_Out1=&IL_avg;//Output average IL after offset correction
    MATH_EMAVG_Multiplier1=_IQ30(0.0080);

    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vp_fb;
    MATH_EMAVG_Out2=&Vp_fb_Avg;
    MATH_EMAVG_Multiplier2=_IQ30(0.008);

        //connect the 2P2Z connections, for the inner Current Loop, Loop1
    CNTL_2P2Z_Ref1 = &Boost_IL_cmd;
    CNTL_2P2Z_Out1 = &DutyA;
    //CNTL_2P2Z_Fdbk1= &IL_raw;//*****Change to this for testing with rev 2 board
    CNTL_2P2Z_Fdbk1= &IL;//Feedback instantaneous IL after offset correction (done in ISR)
    CNTL_2P2Z_Coef1 = &CNTL_2P2Z_Coef8true1.b2;

    PWMDRV_1ch_UpDwnCnt_Duty1 = &DutyA;
    PWMDRV_1ch_UpDwnCnt_Duty2 = &DutyA;

    // Initialize the net variables
    DutyA = _IQ24(0.0);
    Duty4A = _IQ24(0.0);

    IL_avg = _IQ24(0.0);
    IL = _IQ24(0.0);
    IL_raw = _IQ24(0.0);
    Vb_fb_Avg = _IQ24(0.0);
```

1. 打开并检查*SOLAR_DC_DC-DPL-ISR.asm*。 请注意构建 2 下的 _DPL_Init 和 _DPL_ISR 部分。这是所有构建 2 下的全部宏例示分别完成初始化和运行结束的位置。
2. 在*SOLAR_DC_DC-Settings.h*文件中，将递增构建选项选为 2。 然后，为了运行此代码，按照构建 1 中的步骤 2（来自节 3.2.3）至 1（来自节 3.2.6）。 当所有这些步骤完成时，您现在应该在 Main() 的起始位置。

注: 只要您改变了*SOLAR_DC_DCSettings.h*中的递增构建选项，请始终进行“重建全部”操作。

3. 使用 <F8> 键来运行代码，或者使用工具条上的 RUN（运行）按钮，或者使用菜单栏上的 Target → Run（目标 → 运行）。
4. 在观察视图中，添加变量*Boost_IL_cmd*并将其设定为 0.05（采用 Q24 格式表示时等于 838861）。 这个变量设定针对电流控制环路的基准电流指令的大小。
5. 将一个适当的阻性负载跨接在 DC-DC 输出上。 例如，可使用一个额定值为 400W 的 1.0KΩ 电阻器。这在 400V 总线电压上提供一个 160W 负载。
6. 将一个隔离式 DC 电源缓慢施加到电路板上。 当输入电压缓慢上升至 300V 时，监控 DCDC EVM 输出电压。缓慢调整 Boost_IL_cmd 的值（步长 0.01）来将输出电压设定为 385V。使用一个具有电压和电流探测装置的示波器来观察输入电压，输入电流，升压 MOSFET 电压，LLC 初级电流和 PWM 输出。 在使用 300V 升压输入和 1.0kΩ 阻性负载时，当升压输出电压被设定为 373V 时，您应该看到 LLC 输出电压（EVM 输出）的值为 385V。以下是在这个情况下捕捉到的示波器图形。 在这里，Ch1 和 Ch2 显示了升压 PWM 输出。 Ch3 是升压输入电流，而 Ch4 是升压 MOSFET 上的电压。

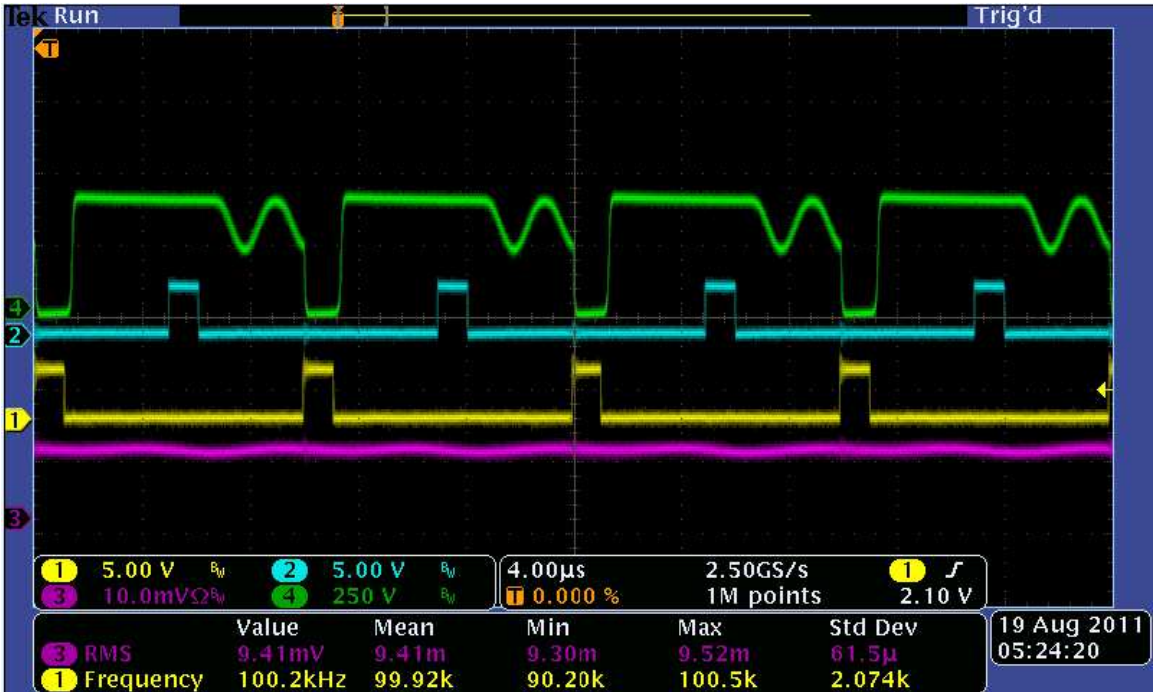


图 **11.** 交错式升压 **DC-DC** 波形

7. 轻微地增加*Boost_IL_cmd*（步长 0.01），并且观察总线电压稳定至一个更高的值。 增加的*Boost_IL_cmd*将增加电流基准信号的振幅，而总线电压将上升。 因此，请采取防护措施并将过压保护阀值设定为一个少于 400V 的值。
8. 按照步骤 11和 12（来自节 3.2.6）来关闭电源并复位 MCU。

3.5 构建 3：具有闭合电压和电流环路的 MPPT DC-DC

3.5.1 目的

这个构建的目的在于从 Code Composer Studio 环境中验证整个 MPPT DC-DC 项目的运行。

3.5.2 概述

图 12显示了这个构建中所用的软件区块。 与图 10中的构建 2 相比，这个构建使用一个标记为*CNTL_2P2Z:2*的额外 2p2z 控制区块。 这是为了执行 MPPT DC-DC 电压环路控制的 2p2z 控制区块的第二例示。 这个电压环路控制器的执行速率为 25kHz，为电流环路执行速率的一半。 这个控制区块的输出驱动电流控制器区块的输入模式*Boost_IL_cmd*。

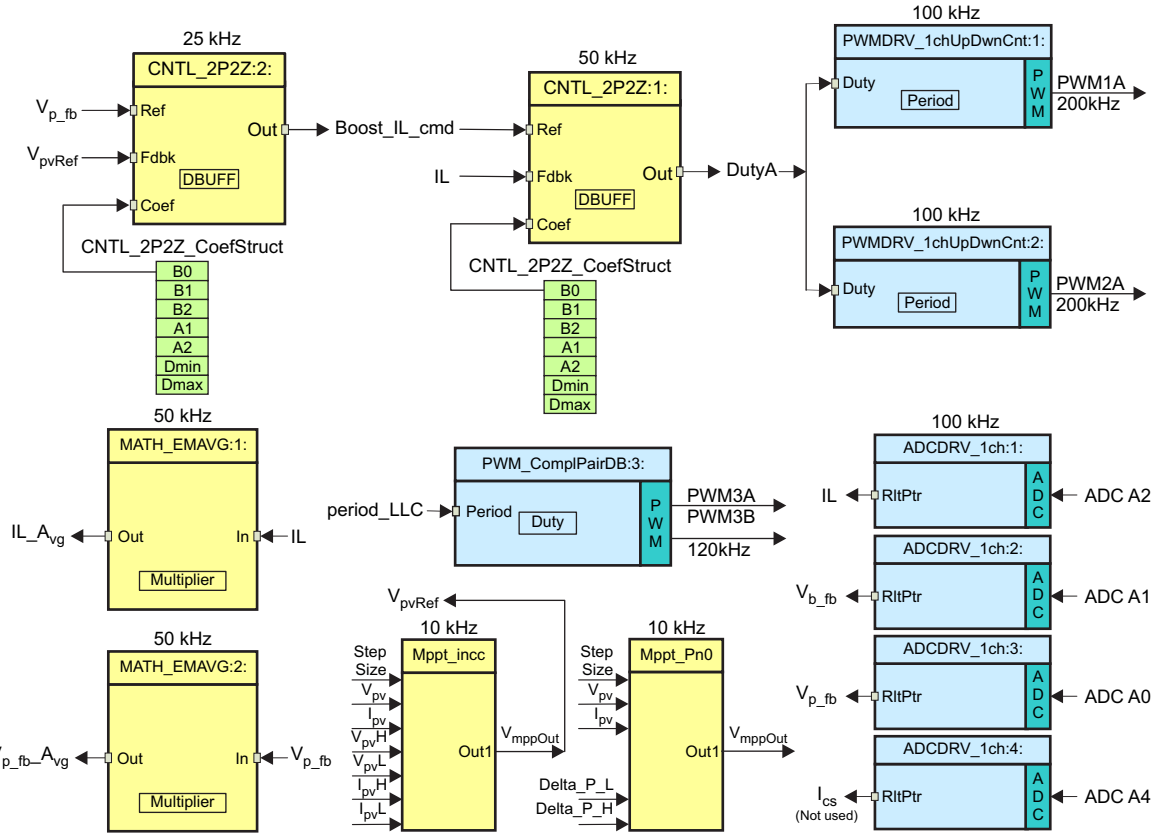


图 **12.** 构建 **3** 软件区块

与电流环路控制器相似，这个电压环路控制器，*CNTL_2P2Z:2*，也要求 5 个控制系数。 这些系数和控制器的钳位输出被存储为一个名为*CNTL_2P2Z_CoefStruct2*的第二结构的元素。 通过直接修改结构*CNTL_2P2Z_CoefStruct2*内的值 B0，B1，B2，A1 和 A2，或者如3.3 节中讨论的那样，通过改变等值 PID 增益，可更改针对这个控制器的系数。

图 12还显示了 2 个执行 2 个不同 MPPT 算法的额外区块，这些算法用于这个 EVM。 此缺省代码设置使用针对 MPPT 的递增电感算法，所以这个 MPPT 区块的输出被连接至电压环路控制器的反馈端子 (Fdbk)。 MPPT 区块运行自一个 10kHz ISR。

3.6 过程

3.6.1 构建和加载项目

按照与构建 1 中的步骤完全一样步骤 1（来自节 3.2.1）至步骤 5（来自节 3.2.2），(3.1 节) 除了那些 6中的步骤，来选择构建 3 选项，而非构建 1。然后按照下面所示完成步骤 6。

确定并检查主文件内初始化代码下面的专门用于构建 3 的代码。这是控制流程中与构建 3 相关的所有软件区块的连接位置。

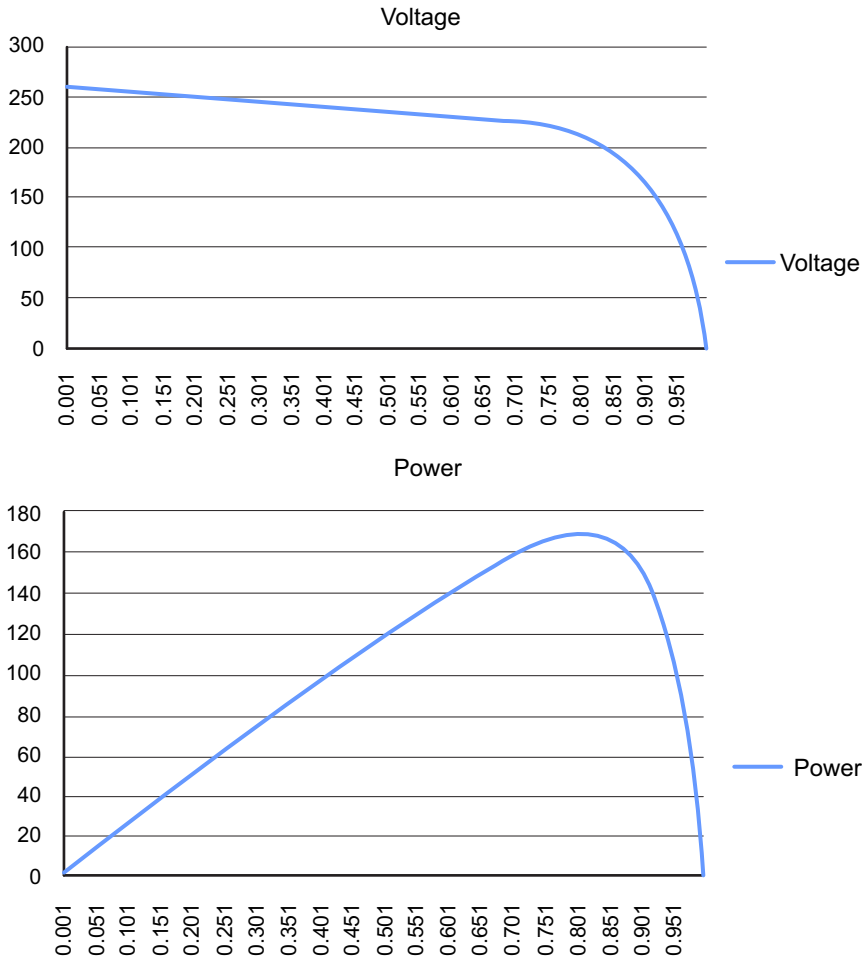
```
//---------------------------------------------------------------------------
#if (INCR_BUILD == 3)   // Closed Current Loop & closed input volt loop dc-dc boost(Panel Volt Output,
                        // or, DC-DC Boost Volt Input)
//---------------------------------------------------------------------------
// Lib Module connection to "nets"
    //-----------------------------------------
    // Connect the PWM Driver input to an input variable, Open Loop System
    ADCDRV_1ch_Rlt1 = &IL_raw;
    ADCDRV_1ch_Rlt2 = &Vb_fb;
    ADCDRV_1ch_Rlt3 = &Vp_fb;
    ADCDRV_1ch_Rlt4 = &Ics;

    // Math_avg block connections - Instance 1
    //MATH_EMAVG_In1=&IL_raw;//*****Change to this for testing with rev 2 board
    MATH_EMAVG_In1=&IL;
    MATH_EMAVG_Out1=&IL_avg;
    MATH_EMAVG_Multiplier1=_IQ30(0.008);//(0.030);

    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vp_fb;//&Vb_fb;
    MATH_EMAVG_Out2=&Vp_fb_Avg;//&Vb_fb_Avg;
    MATH_EMAVG_Multiplier2=_IQ30(0.008);//(0.00025);

        //connect the 2P2Z connections, for the inner Current Loop, Loop1
    CNTL_2P2Z_Ref1 = &Boost_IL_cmd;
    CNTL_2P2Z_Out1 = &DutyA;
    //CNTL_2P2Z_Fdbk1= &IL_raw;//*****Change to this for testing with rev 2 board
    CNTL_2P2Z_Fdbk1= &IL;
    CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;


    //Connect the 2P2Z connections, for the voltage loop for Vin (panel voltage) control, Loop2.
    //*******This volt loop has +ve feedback (compared to the conventional -ve feedback) since for higher Vin, i.e., panel
    //we want higher control output, i.e., in this case higher boost current and vice versa. Thus the net
    //connections below are reversed for feedback and reference terminals**************
    CNTL_2P2Z_Fdbk2 = &VpvRef;//PV panel reference(VpvRef) is connected to the controller feedback net (CNTL_2P2Z_Fdbk2)
    CNTL_2P2Z_Out2 = &Boost_IL_cmd;
    CNTL_2P2Z_Ref2= &Vp_fb;//PV panel feedback(Vp_fb) is connected to the controller reference net (CNTL_2P2Z_Ref2)
    CNTL_2P2Z_Coef2 = &CNTL_2P2Z_CoefStruct2.b2;
```

1. *打开并检查*SOLAR_DC_DC-DPL-ISR.asm。 请注意构建 3 下的 _DPL_Init 和 _DPL_ISR 部分。这是所有构建 3 下的全部宏例示分别完成初始化和运行结束的位置。
2. 在*SOLAR_DC_DCSettings.h*文件中，将递增构建选项选为 3。 然后，为了运行此代码，按照构建 1 中的步骤 2（来自节 3.2.3）至 1（来自节 3.2.6）。 当所有这些步骤完成时，您现在应该在 Main() 的起始位置。

> **注:** 只要您改变了*SOLAR_DC_DC-Settings.h*中的递增构建选项，请始终进行“重建全部”操作。

3. 使用 <F8> 键来运行代码，或者使用工具条上的 RUN（运行）按钮，或者使用菜单栏上的 Target → Run（目标 → 运行）。
4. 在观察视图中，添加 4 个变量 inverter_connected，*Start_DC_DC*，*Vp_fb*和*Vb_fb*。 将最后两个变量（Vp_fb 和 Vb_fb）的 Q 格式设定为 Q24。 这两个变量分别表示升压输入和输出电压。 当施加 PV 电池板仿真器电源并且 MPPT 被打开时，这将在 DC-DC 启动时缓慢增加。 为了首先从 Code Composer Studio 中启动 MPPT（在这个构建下），用户需要修改在下面步骤 6 中解释的代码，并且将程序重新载入到闪存存储器中。 然后，从 Code Composer Studio 观察窗口中，用户将变量*Start_DC_DC*设定为 1，将*inverter_connected*设定为 0。
5. 配置一个太阳能电池板仿真器（200V 至 300V，500W 最大值）来为 EVM 提供输入电源。 配置电池板仿真器来仿真以下的太阳能电池板特性；将其连接至 EVM 输入，但是不要在此时打开电池板电源。

示例电池板仿真器参数		
Voc	开电路电池板电压	260V
Vmpp	针对最大功率点跟踪 (MPPT) 的电池板电压	220V
Impp	针对最大功率点跟踪 (MPPT) 的电池板电流	0.75A
Isc	短路电池板电流	1A



将一个适当的阻性负载连接至 EVM 输出端子上（Vo-R 和 GND 端子）。 如上面的示例所示，如果电池板仿真器被配置为在 MPPT 点上提供 165W 功率，那么选择一个值为 970Ω 的负载电阻器，这样 EVM 输出电压被限制在大约 400V (P = 400*400/970 ≈ 165W)。 只要输出电压下降到低于 350V，更小的电阻器也将起作用。这意味着，针对这个负载设置，可选择的最小电阻器为大约 742Ω (R = 350*350/165 = 742Ω)。 对于这个负载设置，大于 970Ω 的电阻器值将导致输出电压高于 400V。 对于这个 165W 的负载设置，必须通过选择970Ω 的最大电阻器值来防止这个输出过压条件。 针对这个负载设置，建议使用一个功率额定值大于 200W 的电阻器。

6. 从 Code Composer Studio 观察窗口启动 MPPT 算法（在构建 3 内），此时 PV 电池板仿真器电源被施加到 EVM 输入上，要求按照下面所示的那样改变一行代码。 没有这个改变，在以下情况发生时，代码自动启动 MPPT 算法：

   (a) 当施加了最小电池板输入电压时

   (b) 在 GUI 内，用户将变量 inverter_connected 设定为 0。

   当 EVM 出厂时，被载入到闪存存储器内的缺省代码使得这个基于 GUI 的 EVM 以独立模式加电。 因此，只有当用户希望从 Code Composer Studio 观察窗口内启动 MPPT 时，才需要进行这个代码修改。

   打开 Code Composer Studio 项目文件*HV_Solar_DC_DC-Main.c*，并且确定其中的变量 Start_DC_DC 按如下所示被设定为 1 的代码。

```
//---------------------------------------------------------------------
void A3(void)
//---------------------------------------------------------------------
{
    if(INCR_BUILD == 3)
    {
        //When the DC-DC runs without the Inverter, check for min panel voltage before starting MPPT DC-DC.
        //When DC-DC Runs with the Inverter connected, this flag (Start_DC_DC) is not used. Another GPIO (GPIO16)
        //enable signal is then used (activated by the Inverter) to start the DC-DC.
        if(Vp_fb_Avg >= VPV_MIN)
        {

            Start_DC_DC=1;//Start DC-DC MPPT provided LLC PWM is ON(This is checked in the 20kHz SECONDARY ISR)

            //*****************************************************************************************
            //To control the variable "Start_DC_DC" from CCS watch window and run the DC-DC code with external 12V bias (not using PR798 bias su
            //COMMENT OUT the line above. This will allow code to run with MPPT off; Then the user will apply panel volt and start MPPT
            //from CCS watch window by setting Start_DC_DC = 1
            //*****************************************************************************************
        }
        else
        {
            Start_DC_DC = 0;//Do not start DC-DC MPPT
        }

        //if(GpioDataRegs.GPADAT.bit.GPIO16 == 0)
        if(GPIO_status == 0)
        {
            A_Task_Ptr = &A4;
        }
        else
        {
        //--------------------
        //the next time CpuTimer0 'counter' reaches Period value go to A1
        A_Task_Ptr = &A1;
        //--------------------
        }
```

图 13. 实例代码

在闪存存储器内，对这行代码进行注释，保存文件，再编译并且重新载入项目。 按照步骤 1至 3（来自节 3.6.1）操作，并运行代码。 对于观察视图，启用实时模式，然后单击 。 变量*Start_DC_DC*将被设定为 0，而*inverter_connected*将被设定为 1。

7. 使用一个电压计来监控 EVM 输出上的 DC 总线电压。 现在，打开 PV 电池板仿真器，其设置在步骤 5 中进行了描述。 在这个点上，MPPT 将保持关闭，所以，EVM 输出应大约为 260V，而输出电压，在使用 1kΩ 负载电容器时，也将在同一电平上。 在 Code Composer Studio 观察窗口中，将变量*inverter_connected*设定为 0，而将*Start_DC_DC*设定为 1。这启动 MPPT 算法，EVM 输出上升至大约 400V。当 MPPT 打开并且输入电压环路控制器被连接时，电池板输出电压（升压输入电压）将大约为 220V。此 EVM 在 220V 的MPPT 基准电压上传送 165W 的电池板功率（如步骤 5中描述的那样）。

   在这个运行条件下，使用一个示波器来捕捉来自 LLC 级的开关节点电压和变压器初级电流。 这显示在图 14中，在这里，Ch2 表示 LLC 初级开关节点电压，而 Ch1 代表 LLC 初级电流。

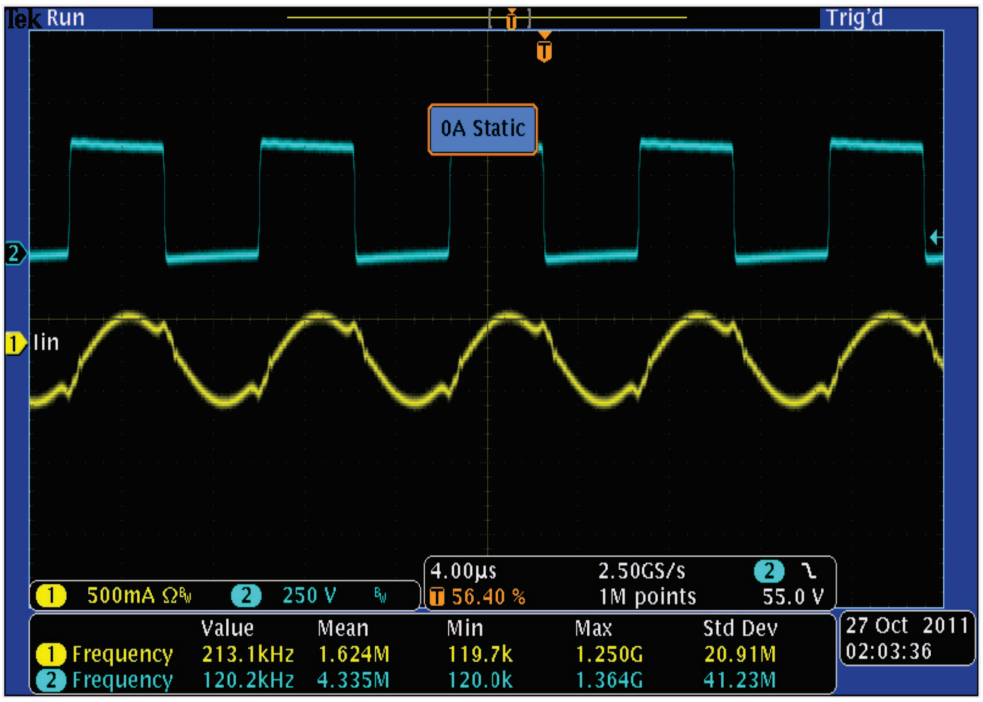


图 **14. LLC** 级波形

现在，使用示波器探测器来捕捉这个运行条件下的升压级 MOSFET 漏源电压和 PV 仿真器电流。 这显示在图 15中，在这里，Ch2 表示升压 MOSFET 漏源电压，而 Ch4 代表电池板电流。

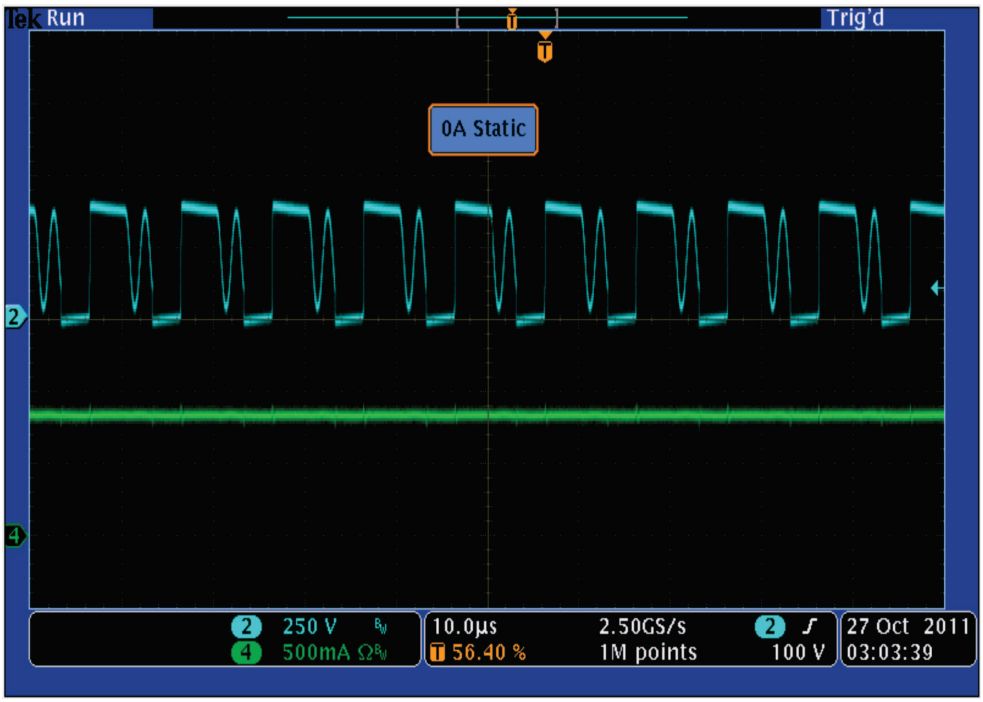


图 **15. DC-DC** 升压级波形

观测观察窗口上的变量。 当 Q 格式被设定为 Q24 时，变量 Vp_fb 应该显示一个大约为 0.4297 的值 (=220/512)。 感测电阻器设定的最大电池板电压大约为 512V，这个值与 3.3V 的最大 ADC 输入电流相对应。 因此，当实际电池板电压为 220Vdc 时，此标准化或标么值将大约为 0.4297。 当 Q 格式被设定为 Q24 时，变量 Vp_fb 应该显示一个大约为 0.7813 的值 (=400/512)。 与 3.3V 的最大 ADC 输入值相对应的，由感测电阻器设定的最大升压输出电压大约为 512V。因此，因此，当实际电池板电压为 400Vdc 时，此标准化或标么值将大约为 0.7813。

8. 按照步骤 1（来自节 3.2.1）至步骤 5（来自节 3.2.2）来关闭电源并复位 MCU。 对于 EVM 的独立运行，取消步骤 6中所执行的代码改变，然后重新编译并将代码重新载入到闪存存储器内。 适当地设置 Piccolo controlCard 跳线（请见 Piccolo controlCard 文档），这样，器件可从闪存引导。

4 参考书目

- www.ti.com/controlsuite：
  - MPPT DC-DC-GUI-QSG - 一个针对使用 GUI 接口的 MPPT DC-DC EVM 快速演示的快速开始指南。

    ...\development_kits\HV_SOLAR_DC_DC\~Docs\QSG_HV_SOLAR_DC_D C_GUI_Rev1.0.pdf
  - MPPT DC-DC_Rel-1.0-HWdevPkg - 一个包含多个与 Piccolo-B 控制器卡电路原理图和 MPPT DC-DC 电路原理图相关的文件的文件夹。

    ...\development_kits\HV_SOLAR_DC-DC\HV_SOLAR_DC-DC_HWDevPkg

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122