User's Guide

F28P55x 与 F28P65x 数字电源应用电子计量库 (E-Metrology Library) 用户指南

摘要

现代电气系统需要高精度计量和全面的电能质量 (PQ) 分析，这两项任务对传统微控制器在实时处理复杂波形和执行计算密集型算法方面构成挑战。本白皮书介绍针对德州仪器 (TI) C2000™ 实时微控制器架构设计的先进计量软件库，该库专门利用这些微控制器集成的 32 位浮点单元 (FPU32)、三角运算单元 (TMU) 和内置的 DSP 功能来克服这些挑战。该库提供一系列丰富的高精度测量，包括真正的 RMS 电压和电流、多相有功、无功和视在功率及能量、功率因数、线路频率、电压骤降/骤升检测，以及基于精确 FFT 的总谐波失真 (THD)，所有测量均使用高精确度 32 位浮点运算进行。TI 的 C2000 MCU 采用这些器件的高效分离式架构，将时间关键型后台数据累积（概念上由 ADC 中断驱动）与复杂的前台计算分开，从而提供了出色的性能。本白皮书重点介绍与通用 MCU 相比，C2000 MCU 在要求严苛的计量和 PQ 应用中的出色性能和精度（例如，原生浮点 FFT 执行和专用 DSP 指令），以及选择 TI C2000 MCU 相较于通用 MCU 的显著优势。

内容

商标

1 简介

电气基础设施的发展（包括智能电网、可再生能源集成和复杂的工业自动化）对电气参数测量的准确性和全面性提出了越来越严格的要求。精确计量（即测量科学）不再只是基本能耗，它现在还涵盖详细的电能质量 (PQ) 分析，这对于维持电网稳定、延长设备寿命以及提高运行效率至关重要。RMS 电压和电流、有功和无功功率、功率因数、频率和谐波失真等参数有助于深入了解电气系统的运行状况和性能。然而，实现稳健的计量和 PQ 分析过程面临重大的技术挑战。现代电气负载通常为非线性，会导致电流和电压波形失真并产生大量谐波。在实时准确处理这些复杂信号的同时，还需执行如快速傅里叶变换 (FFT) 等计算密集型算法进行谐波分析，或通过精确的点积累积计算实现真实 RMS 和功率计算，这对传统微控制器的性能提出了极大挑战。此外，实现计量标准所需的高精度（通常需要使用浮点计算）又增加了一层复杂性，尤其是在资源受限的嵌入式平台上。

2 库声明

该库作为参考代码提供，用于演示德州仪器 (TI) C2000 微控制器上的计量算法概念。该库并非可直接投入生产的解决方案，未经实质性修改和验证，不得用于发布产品。

代码默认以 DEBUG 模式提供，为计量引擎提供软件生成的、数学上完美的正弦波。此默认值为有意设置，因为 DEBUG 模式允许用户验证算法并探索测量输出而无需硬件。但是，DEBUG 模式无法让用户了解系统使用真实模数转换器 (ADC) 输入、真实电流互感器、真实分压器或真实市电信号时的表现。

下面列出了在库可与功率测量系统配合使用之前，需要针对用户硬件进行特定工程工作的领域：

- **ADC 配置** — 必须在系统配置 (**SysConfig**) 菜单中配置片上 ADC，以匹配所用电路板的通道分配、采样率、触发源和采集窗口。默认配置针对特定的 LaunchPad™ 开发套件引脚排列；除非同时修改 *.syscfg* 文件和 *adc_config.h* 文件，否则该配置不适用于定制电路板。
- **模拟前端缩放** — *template.h* 模板中 **PHASE_x_VOLTAGE_SCALE_FACTOR** 和 **PHASE_x_CURRENT_SCALE_FACTOR** 字段的值均为占位符。必须根据实际使用的硬件得出以下值列表：
  - 分压器分压比
  - 电流互感器
  - 分流电阻器
  - 运算放大器 (op-amp) 增益
  - ADC 基准电压

  使用不正确的比例因子会产生在数字上看似合理但不正确的读数。
- **ADC 校准** — 必须按照节 7.2.2 中所述在实际硬件上执行两点校准（直流偏移和增益）。
- **相位校正** — 电流互感器和模拟滤波器级在电压和电流测量路径之间引入相移。该相移直接影响有功功率和功率因数的精度。
- **外部 ADC 支持** — 如果设计使用外部 Σ-Δ 或逐次逼近型寄存器 (SAR) ADC，而不是片上 ADC，则必须完全重写 **ADC_readMetrologyData()** 函数才能与 ADC 驱动程序连接。库架构支持重写此函数，但 *adc_config.c* 文件中的任何内容均无法直接用于不同的 ADC。
- **ISR 时序** — 计量后台任务必须精确地以每秒 SAMPLE_RATE 次采样的速率执行。默认使用 **CPUTimer0** 任务。如果系统使用不同的计时器、直接存储器存取 (DMA) 触发器或 ADC 转换结束中断，则必须相应地更改 *metrology_main_file.c* 文件中的中断服务例程 (ISR) 接线。

所有默认值均未根据真实硬件基准进行验证。这些值的存在是为了让 DEBUG 模式演示生成在数字上合理的输出。将 *template.h* 模板和 *adc_config.h* 文件中的每个数字视为起点，而不是规格。

3 库架构

为了满足实时持续功率监测的严格要求，并防止丢失关键数据，计量库采用高效的分离式处理架构。这种设计将对时间敏感的任务与频率较低、计算量较大的计算任务分离。该设计还可优化资源利用率并保持系统响应能力。该架构大致分为两个主要处理层：

- 后台处理
- 前台处理

3.1 后台处理

后台处理以高频执行，通常与模数转换器 (ADC) 采样率同步。后台处理任务在中断服务例程 (ISR) 内处理。该层的主要职责包括：

- **原始样本采集**：从 ADC 中读取最新的电压和电流样本。
  - 基本预处理：对原始采样值执行最少量的时间关键操作，例如进行直流估算和滤波以消除直流偏置。
- **核心累积**：在指定的测量窗口（通常为一个或多个市电周期）内计算并累积基本点积（例如电压采样平方和、电流采样平方和、电压与电流乘积之和）。这些累积构成了大多数后续计量计算的基础。
- **数据缓冲和同步**：管理数据缓冲区并准备累加的数据，以便将其安全地传输到前台任务。

后台处理的主要设计原则是尽量缩短 ISR 的执行时间，以实现高采样精度，并允许系统响应其他关键中断。

3.2 前台处理

前台处理在主应用程序循环中以较低的频率运行。一旦后台处理累积了一个完整的数据块，就会触发前台任务。该层处理计算量更大的计算任务，包括：

- **检索累加数据**：安全地读取后台任务记录的一致的点积集合和其他相关数据。
- **应用校准**：使用预先确定的校准系数校正原始累积值，以补偿传感器误差及模拟前端的偏差。
- **计算最终参数**：使用校准后的数据计算整套计量参数，例如 RMS 电压和电流、有功功率、无功功率和视在功率、功率因数、频率、THD 及电压骤降和骤升事件。
- **能量累积**：对功率值随时间进行累积计算以得到能耗。
  - 更新结果：为应用提供最终的校准测量值以进行显示、记录或通信。

3.3 数据传输机制

这种分离架构的一个关键要素是后台 ISR 与前台任务之间可靠的数据传输机制。该库采用同步交接技术（通常称为双缓冲）以保持数据完整性。过程如下：

1. 后台 ISR 会不断使用每个样本的数据更新主要累加器集合。
2. 当一个完整的测量块（例如指定数量的样本或市电周期）累积完成后，后台任务会将这些主要累加器的当前值复制到辅助*已记录*变量集合中。
3. 然后，后台任务设置状态标志，向前台任务发出信号，表示已记录的变量中有一个新的一致数据块已就绪。
4. 在主应用循环中运行的前台任务会定期检查此状态标志。检测到该标志被设置时，前台任务就知道*已记录*累加器中存在一组完整且一致的数据。然后，前台任务继续使用此数据完成计算。

4 计量计算

实际计量计算发生在 *meterology_caluclations.c* 文件中。

- 此模块包含实施计量算法的数学函数库。
- 该模块以累积点积、样本计数和校准数据等作为输入。
- 模块会输出计算的电气参数（例如 **calculateRMSVoltage**、**calculateActivePower** 和 **calculateFrequency_PLL**）。

5 配置和数据结构

这些文件定义常量（例如，标称电压、频率或采样率）、用于启用功能的预处理器宏（例如，**HARMONICS_SUPPORT** 和 **NEUTRAL_MONITOR_SUPPORT**），以及保留所有计量相关信息的关键数据结构。

- **metrologyData**：顶层全局结构的实例会保存所有相位、中性、总计和系统状态的所有数据。
- **phaseMetrology** 或 **neutralMetrology**：**metrologyData** 中的结构，包含每个相位和中性线的参数 (params) 和读数 (readings)。
  - **params**：存储配置、后台的中间累积值（如 dotProd 数组）、校准所得的派生值，以及状态变量（例如 PLL 以及骤降和骤升）。
  - **readings**：存储最终计算得出的计量值（RMS、功率、PF 等）。
- **calibrationData** (*metrology_nv_structs.h*)：保存校准系数。

6 校准模块

校准模块管理校准数据的加载、应用和存储（概念性）。

- **Metrology_setup_init**（或 *metrology_setup.c* 中的类似函数）：初始化数据结构并应用初始校准设置，如相位校正。

7 用户特定的配置

用户可以配置计量库用于计算计量参数，以使这些参数满足当前应用的要求。所有面向用户的配置均包含在以下两个文件中：

- *template.h*
- *adc_config.h*

所有其他库文件必须保持不变。

该库支持在编译时选择的两种工作模式：

- DEBUG 模式
- ADC 模式

模式由 *template.h.* 文件中的 DEBUG 预处理器符号控制。

7.1 DEBUG 模式

DEBUG 模式使用软件信号发生器替代真实的 ADC 硬件。DEBUG 模式用于在连接任何硬件之前进行初始启动、算法验证以及验证计量引擎是否生成正确的结果。无需 ADC 硬件或模拟前端。

7.1.1 调试模式的函数

在定义 DEBUG 模式时，**cpuTimer0ISR** 函数调用 **get_debugData()** 函数而不是 **ADC_readMetrologyData()** 函数。**get_debugData()** 函数用于从两个查找表（一个用于电压，一个用于电流）提供预先计算的样本，这些表在启动时由 **debugInit()** 函数填充。

debugInit() 函数填充 voltage[] 和 current[] 数组（每个数组的长度：**SAMPLES_PER_CYCLE** = 50Hz/8000Hz 系统为 160 个样本），使用 C 标准 **sinf()** 函数在一个完整周期 [0, 2*PI] 内以均匀间隔的相位角计算：

- voltage[i] = (V_RMS × SQRT2) / (V_SCALING) × sinf(t)
- current[i] = ((I_RMS × SQRT2) / I_SCALING) × sinf(t - phaseDiff)

其中 t = (i / SAMPLES_PER_CYCLE) × 2*PI，phaseDiff = acosf(POWER_FACTOR)。

在存储之前，样本由 template.h 文件中定义的 **V_SCALING** 和 **I_SCALING** 宏变量进行归一化，以便馈送到计量引擎的值与实际 ADC 样本采用相同的标幺值格式。然后，计量引擎应用校准比例因子（**VscaleFactor** 和 **IscaleFactor**）重构物理单位。在 DEBUG 模式下，这些比例因子在 *metrology_main_file.c* 文件的 **main()** 函数中被覆盖，以匹配 **V_SCALING** 和 **I_SCALING** 参数，从而使显示的读数恰好等于 **V_RMS** 和 **I_RMS** 值。

每个获取 **debugData()** 函数的调用都会在循环查找表中推动一组索引指针（**voltageIndex[]** 和 **currentIndex[]**），因此 ISR 在正确的采样率下会看到连续的周期性波形。

对于多相拓扑，通过在不同的起始位置初始化 **voltageIndex[1]** 和 **voltageIndex[2]**，每个相位相对于相位 A 偏移 **PHASE_INCREMENT** = 53 个样本（相当于 53/160 × 360 = 119.25 度相位间距）。

7.1.2 如何配置 DEBUG 模式参数

所有 DEBUG 模式参数均在 *metrology_main_file.c* 文件顶部的 **#ifdef DEBUG** 块内定义：

- **V_RMS** — 所生成正弦波的均方根 (RMS) 电压，单位为伏特。默认值：120.0.
- **I_RMS** — 所生成正弦波的 RMS 电流，单位为安培。默认值：10.0.
- **POWER_FACTOR** — 仿真负载的功率因数（0.0 至 1.0）。将电压和电流之间的相位角设置为 phaseDiff = acosf(POWER_FACTOR)。默认值：0.5（滞后，60° 移位）。
- **V_SCALING** — 用于归一化所生成样本的电压比例因子。必须与 *template.h* 模板中的 **PHASE_A_VOLTAGE_SCALE_FACTOR** 匹配。默认值：1585.2.
- **I_SCALING** — 用于归一化所生成样本的电流比例因子。必须与 *template.h* 模板中的 **PHASE_A_CURRENT_SCALE_FACTOR** 匹配。默认值：125.0.
- **SAMPLES_PER_CYCLE** — 查找表中每个市电周期的样本数。必须等于 **SAMPLE_RATE | MAINS_NOMINAL_FREQUENCY**。默认值：160 (8000Hz / 50Hz)。
- **PHASE_INCREMENT** — 查找表中相位之间的起始索引偏移。默认值：53.

将谐波注入生成的信号

为了测试总谐波失真 (THD) 计算，可以通过在 *metrology_main_file.c* 文件的 **#ifdef DEBUG** 块内定义 **INJECT_HARMONIC**，将谐波分量注入生成的波形。定义 **INJECT_HARMONIC** 时，在执行 **debugInit()** 函数期间，会向电压和电流查找表添加一个额外的正弦分量：

- voltage[i] ± (V_RMS × SQRT2) / (V_SCALING) × HARMONIC_PERCENT × sinf(HARMONIC_ORDER × t)
- current[i] ± (I_RMS × SQRT2) / (I_SCALING) × HARMONIC_PERCENT × sinf(HARMONIC_ORDER × t - phaseDiff)
- **HARMONIC_ORDER** — 要注入谐波的整数阶数（例如，5 表示以 250Hz 的频率注入 5 次谐波）。默认值：5.
- **HARMONIC_PERCENT** — 注入谐波的幅值占基波幅值的比例（例如，0.50 = 50%）。预期的 THD 结果约等于该值。默认值：0.50.

通过在 **debugInit()** 函数的循环内手动添加更多项，可以注入多个谐波。

7.2 ADC 模式

在 ADC 模式下，该库从通过 **SysConfig** 菜单配置的片上 ADC 外设中读取实际电压和电流样本。存在两种子案例：

- 直接使用片上 ADC
- 用外部 ADC 替换片上 ADC

7.2.1 使用片上 ADC

片上 ADC 完全通过项目中包含的系统配置 (*.syscfg*) 文件进行配置。打开 **SysConfig** 菜单中的 *.syscfg* 文件进行配置：

- ADC 模块选择（默认情况下 ADCA 用于电压，ADCB 用于电流）
- SOC（转换启动）触发源 — 设置为计量 ISR 使用的相同 **CPUTimer0** 周期
- 采样窗口持续时间 — 设置为满足 ADC 输入电路趋稳时间
- 采集窗口和 EOC（转换结束）中断（如果需要）

保存 *.syscfg* 文件后，**SysConfig** 菜单会重新生成 ***board.h*** 头文件和 ***board.c*** 源文件，其中包含该库使用的外设名称（**myADC0_RESULT_BASE**、**myADC0_SOC0** 等）。

打开 ***adc_config.h*** 头文件，并为每个相位的库宏分配生成的名称：

相位 A（始终需要）：

- **ADC_VA_RESULT | ADC_VA_SOC** — 相位 A 电压通道的结果基址和 SOC 编号。
- **ADC_IA_BASE | ADC_IA_RESULT | ADC_IA_SOC** — 相位 A 电流通道的基址、结果基址和 SOC 编号。

相位 B（**TWO_PHASE_SUPPORT**、**THREE_PHASE_SUPPORT** 或 **SPLIT_PHASE_SUPPORT** 需要）：

- **ADC_VB_RESULT | ADC_VB_SOC** — 相位 B 电压通道。
- **ADC_IB_RESULT | ADC_IB_SOC** — 相位 B 电流通道。

相位 C（仅 **THREE_PHASE_SUPPORT** 需要，当前已注释掉，在 *SysConfig* GUI 中添加第三个 ADC 时删除注释 **THREE_PHASE_SUPPORT**）：

- **ADC_VC_RESULT | ADC_VC_SOC** — 相位 C 电压通道。必须首先在 **SysConfig** 菜单中添加第三个 ADC SOC。
- **ADC_IC_RESULT | ADC_IC_SOC** — 相位 C 电流通道。

7.2.2 两点校准

片上 ADC 需要执行两个校准步骤才能生成准确的测量结果：

- 偏移校准
- 增益校准

这些共同形成两点校准，可消除模拟前端的直流偏置并校正由电阻分压器、电流互感器或运算放大器链引入的增益误差。

7.2.2.1 步骤 1 — 偏移校准（零输入点）

交流信号未施加到 ADC 输入端：

1. 调用 **App_startDataCollection()** 函数之前，在启动时调用一次 **ADC_calibrateOffsets()** 函数。
2. 该函数对电压和电流通道上的 1000 个连续 ADC 读数求平均值，以得出直流中点：
   - **ADC_calibrateOffsets();** // 在未连接交流输入时调用
3. 该函数将测得的平均值存储在内部变量 **g_voltageOffset** 和 **g_currentOffset** 中。然后从 **ADC_readNormalized()** 函数的每个后续 ADC 读数中减去这些平均值。对于双极性信号中心位于 Vref/2 (1.65V) 的 12 位 ADC，其理论中点为 2048，但在实际应用中，元件容差会使该中点发生偏移。
4. 如果未调用 **ADC_calibrateOffsets()** 函数，则使用 **ADC_VOLTAGE_DC_OFFSET** 和 **ADC_CURRENT_DC_OFFSET** 的默认值，如 *adc_config.h* 文件中所定义（两者均默认为 2048.0）。

7.2.2.2 步骤 2 — 增益校准（满量程点）

1. 施加已知的交流基准信号，并将库输出与已知值进行比较。
2. 使用以下公式调整 *adc_config.h* 文件中的 **ADC_VOLTAGE_GAIN** 和 **ADC_CURRENT_GAIN**：
   - gain_new = gain_old × (measured_reading / known_reference)
     - 例如，如果库根据已知的 120.0V 基准报告 118.2V RMS：gain_new = 2056.8 × (120.0 / 118.2) = 2087.1
3. 使用 **ADC_CURRENT_GAIN** 对电流重复此操作。 ***ADC_readNormalized()*** 函数按以下公式应用这些增益：
   - normalized = (raw_count - dc_offset) / gain
   - 得出的归一化值是与调试信号发生器输出位于相同范围内的标幺值，然后计量引擎按 VscaleFactor/IscaleFactor（来自 template.h）进行缩放以生成物理单位。

相位校正（用于电流互感器相移）：

电流互感器和模拟滤波器级会在电压和电流通道之间引入微小但可测量的相移，这会直接影响功率因数和功率精度。这可通过 **currentSensorCalibrationData** (*metrology_nv_structs.h*) 中的 **phaseOffset** 字段进行校正。

默认相位校正值在 ***metrology_calibration_defaults.h*** 中定义：

- **DEFAULT_BASE_PHASE_A_CORRECTION** — 相位 A 的相位校正，单位为：微秒数 × **SAMPLE_RATE** × 256。默认值：0.0（适用于 DEBUG 模式和相应的传感器）。对于具有 2.4 微秒滞后的实际 CT，该值为：-2.4e-6 × 8000 × 256。

可接受范围约为 -125 微秒至 +125 微秒。使用精密功率分析仪在单位功率因数下测量实际相位误差，并相应地设置校正。

7.2.3 使用外部 ADC

如果片上 ADC 替换为外部 ADC（例如通过 SPI 连接的 Σ-Δ 器件），则需要进行以下更改：

1. 在 *adc_config.c* 文件中 — 将 **ADC_readMetrologyData()** 函数内的 **ADC_readNormalized()** 函数调用替换为对外部 ADC 驱动程序的调用。该函数必须将归一化的浮点（以零为中心，满量程约为 ±1.0）写入 **workingData→rawVoltageData[PHASE_x]** 和 **workingData→rawCurrentData[PHASE_x]**。归一化约定为：

   normalized = (raw_sample - dc_offset) / gain

   其中，dc_offset 和 gain 是上述相同的两点校准常量，应用于外部 ADC 的原始输出计数。
2. 在 *adc_config.h* 文件中 — 删除或替换 **ADC_VA_RESULT | ADC_VA_SOC** 宏定义，因为 **SysConfig** 菜单中生成的名称不再适用。改为定义驱动程序所需的常量。**ADC_VOLTAGE_DC_OFFSET**、**ADC_CURRENT_DC_OFFSET**、**ADC_VOLTAGE_GAIN** 和 **ADC_CURRENT_GAIN** 保持有效，必须设置为匹配外部 ADC 的分辨率和前端增益。
3. 在 *.syscfg* 文件中 — 如果未使用片上 ADC，请删除 ADCA 和 ADCB 外设实例，以避免消耗存储器和时钟资源。
4. 在 *metrology_main_file.c* 文件中 — 如果外部 ADC 使用数据就绪中断而不是计时器触发器，则将 **ADC_readMetrologyData()** 和 **Metrology_perSampleProcessing()** 函数调用从 **cpuTimer0ISR** 函数移至外部 ADC 中断处理程序中。验证 ISR 是否以与 SAMPLE_RATE 相同的速率执行。
5. 在 *template.h* 文件中 — **PHASE_x_VOLTAGE_SCALE_FACTOR** 和 **PHASE_x_CURRENT_SCALE_FACTOR** 宏仍必须反映外部 ADC 模拟前端的物理缩放，因为计量引擎使用这些宏将归一化样本转换为电压和电流值。

7.3 模式切换

要从 DEBUG 模式切换到实际的 ADC 输入，请注释掉或删除 *metrology_main_file.c* 文件顶部附近的 *#define* DEBUG 行：

```
//#define DEBUG
```

ISR 随后会调用 **ADC_readMetrologyData()** 函数而不是 **get_debugData()** 函数，且覆盖 **#ifdef DEBUG** 宏（该宏作为模式之间切换的挂钩）下 **main()** 函数的校准比例因子不再适用。直接使用 *template.h* 文件中的比例因子。

7.4 THD 算法选择

该库提供三种 THD 计算方法，在编译时通过 *template.h* 模板中的 **HARMONICS_SUPPORT** 和 **USE_GOERTZEL_THD** 宏进行选择：

- 方法 1 — 基于 PLL（未定义 HARMONICS_SUPPORT 宏时的默认值）
- 方法 2 — Goertzel DFT（HARMONICS_SUPPORT 和 USE_GOERTZEL_THD 宏均已定义时）
- 方法 3 — 2048 点实数 FFT（已定义 HARMONICS_SUPPORT 宏但未定义 USE_GOERTZEL_THD 宏时）

7.4.1 方法 1 — 基于 PLL（未定义 HARMONICS_SUPPORT 宏时的默认值）

方法一使用以下关系式计算 THD：

$$THD = \sqrt{RMS\_total^2 - RMS\_fundamental^2} / RMS\_fundamental$$

其中 RMS_total 根据原始波形点积计算得出，RMS_fundamental 使用跟踪 50Hz 或 60Hz 分量的 PLL 提取。这种方法计算量较小，但由于过零触发的积分窗口与 PLL 周期边界之间的窗口不匹配，对于纯净的正弦输入，具有 ≅0.1% 的固有误差下限。结果在 **phaseReadings** 结构的 **voltageTHD** 和 **currentTHD** 字段中返回。

当存储器受到限制或 THD 仅作为定性指标时，请使用方法一。

7.4.2 方法 2 — Goertzel DFT（HARMONICS_SUPPORT 和 USE_GOERTZEL_THD 宏均已定义时）

方法二使用 Goertzel 算法在 1600 个样本窗口（50Hz/8000Hz 采样率时为 10 个周期，60Hz 时为 12 个周期）内计算各个谐波频段（第二次至第 50 次谐波）的 DFT。Goertzel 算法是一种递归滤波器，可为单一频段生成与 FFT 相同的结果，但具有更低的存储器开销且窗口函数不会发生频谱泄漏。然后，通过以下公式计算 THD：

$THD = sqrt(sum(H2^2 + H3^2 + ... + H50^2)) / H1$

其中 H1 是基波振幅，H2...H50 均为谐波振幅。结果在 **phaseReadings** 结构的 **vTHDfft** 和 **iTHDfft** 字段中返回。

此方法是生产计量应用的推荐方法，因为该方法在十到十二个周期内即可稳定下来，并满足 IEC 61000-4-30 标准的 A 类精度要求。

7.4.3 方法 3 — 2048 点实数 FFT（已定义 HARMONICS_SUPPORT 宏但未定义 USE_GOERTZEL_THD 宏时）

方法三使用完整的 2048 点实数 FFT，并在变换之前应用 Hann 窗口。在 50Hz/8000Hz (2048/8000 × 50 = 12.8) 时，FFT 窗口跨度约为 12.8 个周期。在 FFT 之后，根据 IEC 61000-4-7 标准（以每个谐波为中心、5Hz 为一个频段）对谐波进行分组，然后根据分组谐波频段计算 THD。结果将在 **vTHDfft** 和 **iTHDfft** 字段中返回。

该方法提供完整的频谱信息，但与 Goertzel DFT 方法相比，需要消耗明显更多的 RAM（用于实数和虚数 FFT 缓冲器的两个 2048 元素浮点数组）和更长的建立时间。仅当需要 THD 之外的完整频谱分析时，才使用方法三。

7.5 拓扑选择

定义恰好一个拓扑宏以匹配所测量的系统：

- **SINGLE_PHASE_SUPPORT** — 单相，仅相位 A（默认值）。
- **TWO_PHASE_SUPPORT** — 两相，相位 A 和相位 B。
- **THREE_PHASE_SUPPORT** — 三相，相位 A、B 和 C。
- **SPLIT_PHASE_SUPPORT** — 分相，相位 A 和 B 共用中性线。

7.6 启用功能

通过定义或取消定义相应的宏，可以启用或禁用各项测量。以下功能可用且必须在 *template.h* 文件中定义：

- **HARMONICS_SUPPORT** — 基于 FFT 的谐波分析。THD 计算所需。
- **NEUTRAL_MONITOR_SUPPORT** — 中性线电流监测。
- **VRMS_SUPPORT** — RMS 电压。
- **IRMS_SUPPORT** — RMS 电流。
- **ACTIVE_POWER_SUPPORT** — 有功功率。
- **REACTIVE_POWER_SUPPORT** — 无功功率。
- **APPARENT_POWER_SUPPORT** — 视在功率。
- **ACTIVE_ENERGY_SUPPORT** — 有功能量累积。
- **REACTIVE_ENERGY_SUPPORT** — 无功能量累积。
- **POWER_FACTOR_SUPPORT** — 功率因数。
- **POWER_FACTOR_ANGLE_SUPPORT** — 功率因数角度。
- **FREQUENCY_SUPPORT** — 线路频率。
- **SAG_SWELL_SUPPORT** — 电压骤降和骤升检测。
- **FUNDAMENTAL_PARAMETERS** — 启用所有基波（仅限 50Hz 或 60Hz 分量）测量。
- **VOLTAGE_THD_SUPPORT** — 电压 THD。需要 HARMONICS_SUPPORT 宏。
- **CURRENT_THD_SUPPORT** — 电流 THD。需要 HARMONICS_SUPPORT 宏。

定义 **THREE_PHASE_SUPPORT** 宏时，还提供以下各项：

- **LINETOLINE_VOLTAGE_SUPPORT** — 相间电压。
- **AGGREGATE_POWER_FACTOR** — 所有相位的聚合功率因数。
- **CURRENT_VECTOR_SUM** — 电流向量和。
- **PHASE_TO_PHASE_ANGLE_SUPPORT** — 相间角度。

7.7 系统参数

- **SAMPLE_RATE** — ADC 采样率，单位为 Hz。接受的值：2000、2930、3906、4000、5859、6000、6400、7812 和 8000。默认值：8000.
- **MAINS_NOMINAL_FREQUENCY** — 单位为 Hz 的标称市电频率（50 或 60）。默认值：50.
- **MAINS_MIN_FREQUENCY | MAINS_MAX_FREQUENCY** — 过零检测的有效频率范围。默认值：40.0Hz 至 70.0Hz。
- **MAINS_NOMINAL_VOLTAGE** — 单位为 V 的标称 RMS 电压。用作骤降或骤升检测的基准。默认值：230.0V。
- **ENERGY_PULSE_CONSTANT** — 单位为每 kWh 脉冲数的能量脉冲速率。默认值：6400.

7.8 每相位比例因子

比例因子将归一化的 ADC 值转换为物理单位，并且必须与模拟前端硬件（分压器分压比、电流互感器或分流器比率或者运算放大器增益）相匹配。

在 ***template.h*** 文件中找到以下宏：

- **PHASE_A_VOLTAGE_SCALE_FACTOR** — 相位 A 每个归一化 ADC 单位的电压 (V)。
- **PHASE_A_CURRENT_SCALE_FACTOR** — 相位 A 每个归一化 ADC 单位的电流 (A)。
- **PHASE_A_POWER_SCALE_FACTOR** — 设置为电压和电流比例因子的乘积。

相位 B 和相位 C 具有等效的宏（**PHASE_B_*** 和 **PHASE_C_***），这些宏在启用相应拓扑后生效。

可以将直流估算值（**PHASE_A_VOLTAGE_DC_ESTIMATE** 和 **PHASE_A_CURRENT_DC_ESTIMATE**）设置为近似的直流中点，以缩短启动时的滤波器趋稳时间。如果未知，请将直流中点设置为 0.0。

从计算值中减去交流偏移（**PHASE_A_VOLTAGE_AC_OFFSET**、**PHASE_A_CURRENT_AC_OFFSET**、**PHASE_A_ACTIVE_POWER_OFFSET** 和 **PHASE_A_REACTIVE_POWER_OFFSET**）并用于在校准后微调残余测量误差。

8 运行示例

首先，在 Code Composer Studio™ (CCS) 软件中创建并运行该项目：

1. 使用 CPU1_FLASH 构建配置在 CCS 中构建 **energy_metrology_f28p55** 或 **energy_metrology_f28p65** 项目。
2. 项目构建后，右键选择该项目，然后选择 **DEBUG** 按钮。
3. 使用 CCS 调试弹出窗口中的 **RUN** 按钮启动执行。

接下来，当目标在断点处停止或以实时模式运行时，在 CCS 中添加一个新表达式：

- 打开 *Expressions* 窗口 — *Window* → *Show View* → *Expressions*
- 在 *Expressions* 窗格中选择绿色加号图标，然后选择 *Add New Expression*。

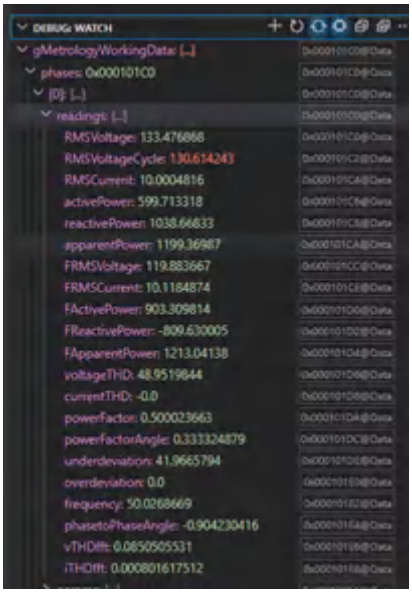

图 8-1. Code Composer Studio (CCS) 软件中的 Expressions 窗口

8.1 访问每个相位的参数

gMetrologyWorkingData 结构包含一个 phases[] 数组，通过 **PHASES** 枚举索引 (**PHASE_ONE** = 0, **PHASE_TWO** = 1, **PHASE_THREE** = 2)。每个相位都有一个 **phaseReadings** 类型的 readings 成员，其中包含最终计算值。

表 8-1. 查看为 phaseReadings 计算的每个相位值

phaseReadings 名称(1)	相位	计算值
gMetrologyWorkingData.phases[0].readings.RMSVoltage	A 相	RMS 电压(2)
gMetrologyWorkingData.phases[0].readings.activePower	A 相	有功功率
gMetrologyWorkingData.phases[0].readings.powerFactor	A 相	功率因数
gMetrologyWorkingData.phases[0].readings.frequency	A 相	频率
基于 PLL（未定义 HARMONICS_SUPPORT 宏时）		
gMetrologyWorkingData.phases[0].readings.voltageTHD	A 相	电压 THD
Goertzel 或 FFT（已定义 HARMONICS_SUPPORT 宏时）		
gMetrologyWorkingData.phases[0].readings.vTHDfft	A 相	电压 THD

(1) 将 [0] 替换为 [1] 表示相位 B，或替换为 [2] 表示相位 C。
(2) 要查看相位 A RMS 电压，必须将 phaseReadings 结构添加到 *Expressions Watch* 窗口。

8.2 访问系统总测量值

系统总计存储在 **gMetrologyWorkingData.totals** 结构中，如下所示：

- gMetrologyWorkingData.totals.readings.activePower
- gMetrologyWorkingData.totals.readings.reactivePower
- gMetrologyWorkingData.totals.readings.apparentPower
- gMetrologyWorkingData.totals.activeEnergy.energy
- gMetrologyWorkingData.totals.reactiveEnergy.energy

8.3 访问相位状态标志

每个相位都有一个状态字段，其中包含 **PHASE_STATUS** 标志的位掩码：**gMetrologyWorkingData.phases[0].status**。

下面列出了 *metrology_defines.h* 文件中定义的有用标志值：

- **PHASE_STATUS_NEW_LOG** (0x0001) — 后台已完成测量窗口；但前台尚未处理该测量值。
- **PHASE_STATUS_V_OVERRANGE** (0x0010) — 电压 ADC 输入达到终点限位（检测到削波）。
- **PHASE_STATUS_I_OVERRANGE** (0x0020) — 电流 ADC 输入达到终点限位。
- **PHASE_STATUS_ZERO_CROSSING_MISSED** (0x4000) — 预期的过零未在有效窗口内发生（频率超出范围或信号丢失）。

要检查是否已设置标志，请以十六进制格式显示状态字段或使用按位 **AND** 表达式：如果错过了过零，则 **gMetrologyWorkingData.phases[0].status** 和 0x4000 均不为零。

备注

要在不停止的情况下进行真正的实时监测，请启用 CCS 连续刷新模式（需要具有实时存储器访问功能的 XDS 调试探针）。启用连续刷新模式：

1. 在 *Expressions* 窗口中右键选择 → *Continuous Refresh* → *Enable*。
   - 在 CPU 继续运行时，这会定期轮询目标存储器。
   - 启用此功能会增加总线流量，并可能影响实时性能。因此，请谨慎使用此模式。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月